OrCAD PSpice A/D

How to use this online manual

New! How to print this online manual

Welcome to OrCAD

Overview

Commands

Analog devices

Digital devices

Customizing device equations

Glossary

Index
# Contents

## How to Use This Online Manual
- How to print this online manual ........................................... xiv
- Welcome to OrCAD ............................................................. xv
- Overview .............................................................................. xvi
- Typographical conventions .................................................. xvi
- Command syntax formats .................................................... xvii
- Numeric value conventions .................................................. xviii
- Numeric expression conventions .......................................... xix

## Command line options
- for OrCAD applications ..................................................... xxii
- Command files ..................................................................... xxii
- Creating and editing command files ..................................... xxii
- Log files ............................................................................. xxiii
- Editing log files ................................................................... xxiv
- Simulation command line specification format ..................... xxv
- Simulation command line options ....................................... xxvi
- Specifying simulation command line options ....................... xxvii

## Commands
- Command reference
  - for PSpice and PSpice A/D .................................................. 30
  - .AC (AC analysis) ............................................................ 32
  - .ALIASES, .ENDALIASES (aliases and endaliases) ............. 33
  - .DC (DC analysis) ............................................................ 34
    - Linear sweep ................................................................ 35
    - Logarithmic sweep ...................................................... 35
    - Nested sweep ................................................................ 36
  - .DISTRIBUTION (user-defined distribution) ....................... 37
    - Deriving updated parameter values ................................ 37
      - Usage example .......................................................... 38
  - .END (end of circuit) ....................................................... 39
  - .EXTERNAL (external port) ............................................... 40
  - .FOUR (Fourier analysis) ............................................... 41
  - .FUNC (function) ............................................................ 42
  - .IC (initial bias point condition) ....................................... 43
  - .INC (include file) .......................................................... 44
  - .LIB (library file) .......................................................... 45
  - .LOADBIAS (load bias point file) ..................................... 46
  - .MC (Monte Carlo analysis) ............................................ 47
  - .MODEL (model definition) ............................................. 50
    - Parameters for setting temperature ................................ 53
      - Model parameters for device temperature ..................... 53
    - Examples ................................................................. 53

## Typographical conventions

## Command syntax formats

## Numeric value conventions

## Numeric expression conventions

## Command line options for OrCAD applications

## Command files

## Creating and editing command files

## Log files

## Editing log files

## Simulation command line specification format

## Simulation command line options

## Specifying simulation command line options
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Special considerations.</td>
<td>54</td>
</tr>
<tr>
<td>.NODESET (set approximate node voltage for bias point)</td>
<td>55</td>
</tr>
<tr>
<td>.NOISE (noise analysis)</td>
<td>56</td>
</tr>
<tr>
<td>.OP (bias point)</td>
<td>58</td>
</tr>
<tr>
<td>.OPTIONS (analysis options)</td>
<td>59</td>
</tr>
<tr>
<td>Flag options</td>
<td>59</td>
</tr>
<tr>
<td>Option with a name as its value</td>
<td>60</td>
</tr>
<tr>
<td>Numerical options with their default values</td>
<td>61</td>
</tr>
<tr>
<td>PSpice A/D digital simulation condition messages</td>
<td>63</td>
</tr>
<tr>
<td>.PARAM (parameter)</td>
<td>65</td>
</tr>
<tr>
<td>.PLOT (plot)</td>
<td>66</td>
</tr>
<tr>
<td>.PRINT (print)</td>
<td>68</td>
</tr>
<tr>
<td>.PROBE (Probe)</td>
<td>69</td>
</tr>
<tr>
<td>DC Sweep and transient analysis output variables</td>
<td>70</td>
</tr>
<tr>
<td>Multiple-terminal devices</td>
<td>71</td>
</tr>
<tr>
<td>AC analysis</td>
<td>73</td>
</tr>
<tr>
<td>Noise analysis</td>
<td>74</td>
</tr>
<tr>
<td>.SAVEBIAS (save bias point to file)</td>
<td>75</td>
</tr>
<tr>
<td>Usage examples</td>
<td>76</td>
</tr>
<tr>
<td>.SENS (sensitivity analysis)</td>
<td>78</td>
</tr>
<tr>
<td>.STEP (parametric analysis)</td>
<td>79</td>
</tr>
<tr>
<td>Usage examples</td>
<td>81</td>
</tr>
<tr>
<td>.STIMLIB (stimulus library file)</td>
<td>82</td>
</tr>
<tr>
<td>.STIMULUS (stimulus)</td>
<td>83</td>
</tr>
<tr>
<td>.SUBCKT (subcircuit)</td>
<td>84</td>
</tr>
<tr>
<td>.ENDS (end subcircuit)</td>
<td>84</td>
</tr>
<tr>
<td>Usage examples</td>
<td>86</td>
</tr>
<tr>
<td>.TEMP (temperature)</td>
<td>87</td>
</tr>
<tr>
<td>.TEXT (text parameter)</td>
<td>88</td>
</tr>
<tr>
<td>.TF (transfer)</td>
<td>89</td>
</tr>
<tr>
<td>.TRAN (transient analysis)</td>
<td>90</td>
</tr>
<tr>
<td>.VECTOR (digital output)</td>
<td>92</td>
</tr>
<tr>
<td>.WATCH (watch analysis results)</td>
<td>94</td>
</tr>
<tr>
<td>.WCASE (sensitivity/worst-case analysis)</td>
<td>95</td>
</tr>
<tr>
<td>* (comment)</td>
<td>99</td>
</tr>
<tr>
<td>; (in-line comment)</td>
<td>100</td>
</tr>
<tr>
<td>+ (line continuation)</td>
<td>101</td>
</tr>
<tr>
<td>Differences between PSpice and Berkeley SPICE2</td>
<td>102</td>
</tr>
<tr>
<td><strong>Analog devices</strong></td>
<td></td>
</tr>
<tr>
<td>Analog devices</td>
<td>106</td>
</tr>
<tr>
<td>Device types</td>
<td>107</td>
</tr>
<tr>
<td>Analog device summary</td>
<td>107</td>
</tr>
<tr>
<td>GaAsFET</td>
<td>110</td>
</tr>
<tr>
<td>Capture parts</td>
<td>111</td>
</tr>
<tr>
<td>Setting operating temperature</td>
<td>111</td>
</tr>
<tr>
<td>Model parameters</td>
<td>112</td>
</tr>
<tr>
<td>GaAsFET model parameters for all levels</td>
<td>112</td>
</tr>
<tr>
<td>GaAsFET model parameters specific to model levels</td>
<td>113</td>
</tr>
<tr>
<td>Contents</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Auxiliary model parameters BTRK, DVT, and DVTT</td>
<td>116</td>
</tr>
<tr>
<td>GaAsFET equations</td>
<td>117</td>
</tr>
<tr>
<td>GaAsFET equations for DC current: all levels</td>
<td>117</td>
</tr>
<tr>
<td>GaAsFET equations for DC current: specific to model levels</td>
<td>118</td>
</tr>
<tr>
<td>GaAsFET equations for capacitance</td>
<td>123</td>
</tr>
<tr>
<td>GaAsFET equations for temperature effect</td>
<td>125</td>
</tr>
<tr>
<td>GaAsFET equations for noise</td>
<td>126</td>
</tr>
<tr>
<td>References</td>
<td>127</td>
</tr>
<tr>
<td>Capacitor</td>
<td>128</td>
</tr>
<tr>
<td>Capture parts</td>
<td>129</td>
</tr>
<tr>
<td>Breakout parts</td>
<td>129</td>
</tr>
<tr>
<td>Capacitor model parameters</td>
<td>130</td>
</tr>
<tr>
<td>Capacitor equations</td>
<td>130</td>
</tr>
<tr>
<td>Capacitor value formula</td>
<td>130</td>
</tr>
<tr>
<td>Capacitor equation for noise</td>
<td>130</td>
</tr>
<tr>
<td>Diode</td>
<td>131</td>
</tr>
<tr>
<td>Capture parts</td>
<td>132</td>
</tr>
<tr>
<td>Setting operating temperature</td>
<td>132</td>
</tr>
<tr>
<td>Diode model parameters</td>
<td>133</td>
</tr>
<tr>
<td>Diode equations</td>
<td>134</td>
</tr>
<tr>
<td>Diode equations for DC current</td>
<td>134</td>
</tr>
<tr>
<td>Diode equations for capacitance</td>
<td>134</td>
</tr>
<tr>
<td>Diode equations for noise</td>
<td>135</td>
</tr>
<tr>
<td>References</td>
<td>135</td>
</tr>
<tr>
<td>Diode equations for temperature effects</td>
<td>135</td>
</tr>
<tr>
<td>Voltage-controlled voltage source</td>
<td>136</td>
</tr>
<tr>
<td>Voltage-controlled current source</td>
<td>136</td>
</tr>
<tr>
<td>Basic SPICE polynomial expressions (POLY)</td>
<td>138</td>
</tr>
<tr>
<td>Basic controlled source properties</td>
<td>138</td>
</tr>
<tr>
<td>Implementation examples</td>
<td>139</td>
</tr>
<tr>
<td>Current-controlled current source</td>
<td>141</td>
</tr>
<tr>
<td>Current-controlled voltage source</td>
<td>141</td>
</tr>
<tr>
<td>Basic SPICE polynomial expressions (POLY)</td>
<td>141</td>
</tr>
<tr>
<td>Independent current source &amp; stimulus</td>
<td>142</td>
</tr>
<tr>
<td>Independent voltage source &amp; stimulus</td>
<td>142</td>
</tr>
<tr>
<td>Independent current source &amp; stimulus (EXP)</td>
<td>144</td>
</tr>
<tr>
<td>Independent current source and stimulus exponential waveform formulas</td>
<td>144</td>
</tr>
<tr>
<td>Independent current source &amp; stimulus (PULSE)</td>
<td>145</td>
</tr>
<tr>
<td>Independent current source and stimulus pulse waveform formulas</td>
<td>146</td>
</tr>
<tr>
<td>Independent current source &amp; stimulus (PWL)</td>
<td>147</td>
</tr>
<tr>
<td>Independent current source &amp; stimulus (SFFM)</td>
<td>150</td>
</tr>
<tr>
<td>Independent current source &amp; stimulus (SIN)</td>
<td>151</td>
</tr>
<tr>
<td>Independent current source and stimulus sinusoidal waveform formulas</td>
<td>152</td>
</tr>
<tr>
<td>Junction FET</td>
<td>153</td>
</tr>
<tr>
<td>Capture parts</td>
<td>154</td>
</tr>
<tr>
<td>Setting operating temperature</td>
<td>154</td>
</tr>
<tr>
<td>Contents</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Model parameters</td>
<td>155</td>
</tr>
<tr>
<td>JFET equations</td>
<td>156</td>
</tr>
<tr>
<td>JFET equations for DC current</td>
<td>157</td>
</tr>
<tr>
<td>JFET equations for capacitance</td>
<td>158</td>
</tr>
<tr>
<td>JFET equations for temperature effects</td>
<td>159</td>
</tr>
<tr>
<td>JFET equations for noise</td>
<td>159</td>
</tr>
<tr>
<td>Reference</td>
<td>159</td>
</tr>
<tr>
<td>Inductor coupling (and magnetic core)</td>
<td>160</td>
</tr>
<tr>
<td>Transmission line coupling</td>
<td>160</td>
</tr>
<tr>
<td>Inductor coupling</td>
<td>161</td>
</tr>
<tr>
<td>Capture parts</td>
<td>163</td>
</tr>
<tr>
<td>Breakout parts</td>
<td>163</td>
</tr>
<tr>
<td>Inductor coupling: Jiles-Atherton model</td>
<td>165</td>
</tr>
<tr>
<td>Inductor coupling model parameters</td>
<td>165</td>
</tr>
<tr>
<td>Including air-gap effects in the inductor coupling model</td>
<td>166</td>
</tr>
<tr>
<td>Getting core inductor coupling model values</td>
<td>167</td>
</tr>
<tr>
<td>Transmission line coupling</td>
<td>167</td>
</tr>
<tr>
<td>Example</td>
<td>168</td>
</tr>
<tr>
<td>Lossy lines</td>
<td>168</td>
</tr>
<tr>
<td>References</td>
<td>169</td>
</tr>
<tr>
<td>Inductor</td>
<td>170</td>
</tr>
<tr>
<td>Capture parts</td>
<td>171</td>
</tr>
<tr>
<td>Breakout parts</td>
<td>172</td>
</tr>
<tr>
<td>Inductor equations</td>
<td>173</td>
</tr>
<tr>
<td>Inductance value formula</td>
<td>173</td>
</tr>
<tr>
<td>Inductor equation for noise</td>
<td>173</td>
</tr>
<tr>
<td>Inductor model parameters</td>
<td>173</td>
</tr>
<tr>
<td>Inductor</td>
<td>170</td>
</tr>
<tr>
<td>MOSFET</td>
<td>174</td>
</tr>
<tr>
<td>Capture parts</td>
<td>177</td>
</tr>
<tr>
<td>Setting operating temperature</td>
<td>177</td>
</tr>
<tr>
<td>MOSFET model parameters</td>
<td>178</td>
</tr>
<tr>
<td>For all model levels</td>
<td>178</td>
</tr>
<tr>
<td>Model levels 1, 2, and 3</td>
<td>178</td>
</tr>
<tr>
<td>Model level 4</td>
<td>178</td>
</tr>
<tr>
<td>Model level 5 (EKV version 2.6)</td>
<td>179</td>
</tr>
<tr>
<td>Model level 6 (BSIM3 version 2.0).</td>
<td>181</td>
</tr>
<tr>
<td>Model level 7 (BSIM3 version 3.1).</td>
<td>181</td>
</tr>
<tr>
<td>MOSFET model parameters</td>
<td>184</td>
</tr>
<tr>
<td>MOSFET Equations</td>
<td>198</td>
</tr>
<tr>
<td>MOSFET equations for DC current</td>
<td>199</td>
</tr>
<tr>
<td>MOSFET equations for capacitance</td>
<td>200</td>
</tr>
<tr>
<td>MOSFET equations for temperature effects</td>
<td>201</td>
</tr>
<tr>
<td>MOSFET equations for noise</td>
<td>202</td>
</tr>
<tr>
<td>References</td>
<td>203</td>
</tr>
<tr>
<td>Bipolar transistor</td>
<td>204</td>
</tr>
<tr>
<td>Capture parts</td>
<td>205</td>
</tr>
<tr>
<td>Setting operating temperature</td>
<td>205</td>
</tr>
<tr>
<td>Bipolar transistor model parameters</td>
<td>206</td>
</tr>
<tr>
<td>Distribution of the CJC capacitance</td>
<td>208</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Bipolar transistor equations</td>
<td>209</td>
</tr>
<tr>
<td>Bipolar transistor equations for DC current</td>
<td>210</td>
</tr>
<tr>
<td>Bipolar transistor equations for capacitance</td>
<td>211</td>
</tr>
<tr>
<td>Bipolar transistor equations for quasi-saturation effect</td>
<td>212</td>
</tr>
<tr>
<td>Bipolar transistor equations for temperature effect</td>
<td>213</td>
</tr>
<tr>
<td>Bipolar transistor equations for noise</td>
<td>214</td>
</tr>
<tr>
<td>References</td>
<td>214</td>
</tr>
<tr>
<td>Resistor</td>
<td>215</td>
</tr>
<tr>
<td>Capture parts</td>
<td>215</td>
</tr>
<tr>
<td>Breakout parts</td>
<td>216</td>
</tr>
<tr>
<td>Resistor model parameters</td>
<td>217</td>
</tr>
<tr>
<td>Resistor equations</td>
<td>218</td>
</tr>
<tr>
<td>Resistor value formulas</td>
<td>218</td>
</tr>
<tr>
<td>Resistor equation for noise</td>
<td>218</td>
</tr>
<tr>
<td>Voltage-controlled switch</td>
<td>219</td>
</tr>
<tr>
<td>Capture parts</td>
<td>220</td>
</tr>
<tr>
<td>Ideal switches</td>
<td>220</td>
</tr>
<tr>
<td>Voltage-controlled switch model parameters</td>
<td>220</td>
</tr>
<tr>
<td>Special considerations</td>
<td>220</td>
</tr>
<tr>
<td>Voltage-controlled switch equations</td>
<td>221</td>
</tr>
<tr>
<td>Voltage-controlled switch equations for switch resistance</td>
<td>222</td>
</tr>
<tr>
<td>Voltage-controlled switch equation for noise</td>
<td>222</td>
</tr>
<tr>
<td>Transmission line</td>
<td>223</td>
</tr>
<tr>
<td>Ideal line</td>
<td>224</td>
</tr>
<tr>
<td>Lossy line</td>
<td>225</td>
</tr>
<tr>
<td>Capture parts</td>
<td>226</td>
</tr>
<tr>
<td>Ideal and lossy transmission lines</td>
<td>226</td>
</tr>
<tr>
<td>Coupled transmission lines</td>
<td>227</td>
</tr>
<tr>
<td>Simulating coupled lines</td>
<td>228</td>
</tr>
<tr>
<td>Simulation considerations</td>
<td>228</td>
</tr>
<tr>
<td>Transmission line model parameters</td>
<td>229</td>
</tr>
<tr>
<td>References</td>
<td>230</td>
</tr>
<tr>
<td>Independent voltage source &amp; stimulus</td>
<td>231</td>
</tr>
<tr>
<td>Current-controlled switch</td>
<td>232</td>
</tr>
<tr>
<td>Capture parts</td>
<td>233</td>
</tr>
<tr>
<td>Ideal switches</td>
<td>233</td>
</tr>
<tr>
<td>Current-controlled switch model parameters</td>
<td>234</td>
</tr>
<tr>
<td>Special considerations</td>
<td>234</td>
</tr>
<tr>
<td>Current-controlled switch equations</td>
<td>234</td>
</tr>
<tr>
<td>Current-controlled switch equations for switch resistance</td>
<td>235</td>
</tr>
<tr>
<td>Current-controlled switch equation for noise</td>
<td>235</td>
</tr>
<tr>
<td>Subcircuit instantiation</td>
<td>236</td>
</tr>
<tr>
<td>IGBT</td>
<td>237</td>
</tr>
<tr>
<td>Capture parts</td>
<td>238</td>
</tr>
<tr>
<td>Setting operating temperature</td>
<td>238</td>
</tr>
<tr>
<td>IGBT device parameters</td>
<td>239</td>
</tr>
<tr>
<td>IGBT model parameters</td>
<td>240</td>
</tr>
<tr>
<td>IGBT equations</td>
<td>241</td>
</tr>
<tr>
<td>IGBT equations for DC current</td>
<td>242</td>
</tr>
</tbody>
</table>
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IGBT equations for capacitance</td>
<td>243</td>
</tr>
<tr>
<td>References</td>
<td>244</td>
</tr>
<tr>
<td><strong>Digital devices</strong></td>
<td></td>
</tr>
<tr>
<td>Digital device summary</td>
<td>246</td>
</tr>
<tr>
<td>Digital primitive summary</td>
<td>247</td>
</tr>
<tr>
<td>General digital primitive format</td>
<td>250</td>
</tr>
<tr>
<td>Timing models</td>
<td></td>
</tr>
<tr>
<td>Treatment of unspecified propagation delays</td>
<td>252</td>
</tr>
<tr>
<td>Treatment of unspecified timing constraints</td>
<td>253</td>
</tr>
<tr>
<td>Gates</td>
<td></td>
</tr>
<tr>
<td>Standard gates</td>
<td>254</td>
</tr>
<tr>
<td>Standard gate timing model parameters</td>
<td>255</td>
</tr>
<tr>
<td>Tristate gates</td>
<td>257</td>
</tr>
<tr>
<td>Tristate gate types</td>
<td>258</td>
</tr>
<tr>
<td>Tristate gate timing model parameters</td>
<td>260</td>
</tr>
<tr>
<td>Bidirectional transfer gates</td>
<td>261</td>
</tr>
<tr>
<td>Flip-flops and latches</td>
<td></td>
</tr>
<tr>
<td>Initialization</td>
<td>264</td>
</tr>
<tr>
<td>Timing violations</td>
<td>264</td>
</tr>
<tr>
<td>Edge-triggered flip-flops</td>
<td>265</td>
</tr>
<tr>
<td>Edge-triggered flip-flop timing model parameters</td>
<td>267</td>
</tr>
<tr>
<td>Edge-triggered flip-flop truth tables DFF and JKFF</td>
<td>268</td>
</tr>
<tr>
<td>Edge-triggered flip-flop truth tables DFFDE and JKFFDE</td>
<td>269</td>
</tr>
<tr>
<td>Gated latch</td>
<td>270</td>
</tr>
<tr>
<td>Gated latch truth tables</td>
<td>272</td>
</tr>
<tr>
<td>Pullup and pulldown</td>
<td>273</td>
</tr>
<tr>
<td>Delay line</td>
<td>274</td>
</tr>
<tr>
<td>Programmable logic array</td>
<td>275</td>
</tr>
<tr>
<td>Read only memory</td>
<td>279</td>
</tr>
<tr>
<td>Random access read-write memory</td>
<td>283</td>
</tr>
<tr>
<td>Multi-bit A/D and D/A converter</td>
<td>286</td>
</tr>
<tr>
<td>Multi-bit analog-to-digital converter</td>
<td>287</td>
</tr>
<tr>
<td>Multi-bit digital-to-analog converter</td>
<td>289</td>
</tr>
<tr>
<td>Behavioral primitives</td>
<td></td>
</tr>
<tr>
<td>Logic expression</td>
<td>291</td>
</tr>
<tr>
<td>Pin-to-pin delay</td>
<td>292</td>
</tr>
<tr>
<td>Constraint checker</td>
<td>295</td>
</tr>
<tr>
<td>Stimulus devices</td>
<td>304</td>
</tr>
<tr>
<td>Stimulus generator</td>
<td></td>
</tr>
<tr>
<td>Time units</td>
<td>310</td>
</tr>
<tr>
<td>Stimulus generator examples</td>
<td>311</td>
</tr>
<tr>
<td>File stimulus</td>
<td></td>
</tr>
<tr>
<td>Stimulus file format</td>
<td>312</td>
</tr>
<tr>
<td>Transition format</td>
<td>317</td>
</tr>
<tr>
<td>File stimulus device</td>
<td>317</td>
</tr>
<tr>
<td>Input/output model</td>
<td></td>
</tr>
<tr>
<td>Input/output model parameters</td>
<td>319</td>
</tr>
<tr>
<td>Digital/analog interface devices</td>
<td>322</td>
</tr>
<tr>
<td>Input/output model parameters</td>
<td>322</td>
</tr>
<tr>
<td>Digital/analog interface devices</td>
<td>324</td>
</tr>
</tbody>
</table>
Contents

Digital input (N device) .................................................. 324
  Digital input model parameters .................................. 325
Digital output (O Device) ............................................. 328
  Digital output model parameters ................................. 328
Digital model libraries ............................................. 332
  7400-series TTL and CMOS library files ....................... 333
  4000-series CMOS library ........................................ 333
  Programmable array logic devices ............................. 334

Customizing device equations
Introduction to Device Equations ................................. 336
Making device model changes ..................................... 337
  Changing a parameter name ..................................... 338
  Giving a parameter an alias .................................... 338
  Adding a parameter .............................................. 338
  Changing the device equations .................................. 339
    Functional subsections of the device source file ....... 340
  Adding a new device ............................................. 341
  Specifying new internal device structure ................... 342
    Example ........................................................ 342
    Procedure .................................................... 343
Recompiling and linking the
  Device Equations option ........................................ 345
  Personalizing your DLL ........................................ 345
Simulating with the Device Equations option ................ 346
  Selecting which models to use from a Device Equations DLL 346

Glossary

Index
# How to Use This Online Manual

Click this toolbar button or book icon... | To do this...
---|---
| Go back and forth between pages.
| Go back and forth between views.
| Go back to the beginning of the section.
| Go back to the beginning of the chapter.
| Go to the Commands chapter. (Other chapters have similar icons.)
| Go to the Index.
| Go to the Glossary.
| Go to the Contents.
How to print this online manual

You can print any portion of this manual, or the entire book, to keep as a printed reference. The pages are designed to print on 8.5”-by-11” paper, with a left margin wide enough to punch holes for use in a binder.

To print this manual

1. In Acrobat Reader, from the File menu, choose Print.
2. Under Print Range, choose one of the following:
   - All Pages, if you want to print the entire book
   - Current Page, if you want to print the current page only
   - Pages, if you want to print a range of pages (such as a chapter—see the table below)
3. Click OK.

<table>
<thead>
<tr>
<th>To print this chapter...</th>
<th>Print this range of pages...</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to Use This Online Manual</td>
<td>xiii to xxviii</td>
</tr>
<tr>
<td>Commands</td>
<td>29 to 103</td>
</tr>
<tr>
<td>Analog devices</td>
<td>105 to 244</td>
</tr>
<tr>
<td>Digital devices</td>
<td>245 to 334</td>
</tr>
<tr>
<td>Customizing device equations</td>
<td>335 to 346</td>
</tr>
<tr>
<td>Glossary</td>
<td>347 to 351</td>
</tr>
<tr>
<td>Index</td>
<td>353 to the last page of this manual</td>
</tr>
</tbody>
</table>
Welcome to OrCAD

OrCAD® offers a total solution for your core design tasks: schematic- and VHDL-based design entry; FPGA and CPLD design synthesis; digital, analog, and mixed-signal simulation; and printed circuit board layout. What's more, OrCAD’s products are a suite of applications built around an engineer's design flow—not just a collection of independently developed point tools. PSpice and PSpice A/D are just one element in OrCAD's total solution design flow.

Welcome to OrCAD. With OrCAD's products, you'll spend less time dealing with the details of tool integration, devising workarounds, and manually entering data to keep files in sync. Our products will help you build better products, faster, and at lower cost.
Overview

This manual contains the reference material needed when working with special circuit analyses in PSpice A/D.

Included in this manual are detailed command descriptions, start-up option definitions, and a list of supported devices in the digital and analog device libraries.

This manual has comprehensive reference material for all of the PSpice circuit analysis applications, which include:

- PSpice A/D
- PSpice A/D Basics
- PSpice

This manual assumes that you are familiar with Microsoft Windows (NT or 95), including how to use icons, menus and dialog boxes. It also assumes you have a basic understanding about how Windows manages applications and files to perform routine tasks, such as starting applications and opening and saving your work. If you are new to Windows, please review your Microsoft Windows User’s Guide.

Typographical conventions

This manual generally follows the conventions used in the Microsoft Windows User’s Guide. Procedures for performing an operation are generally numbered with the following typographical conventions.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Examples</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>monospace font</td>
<td>mydiodes.slb</td>
<td>Library files and file names.</td>
</tr>
<tr>
<td>key cap or letter</td>
<td>Press enter</td>
<td>A specific key or key stroke on the keyboard.</td>
</tr>
<tr>
<td>monospace font</td>
<td>Type VAC...</td>
<td>Output produced by a printer and commands/text entered from the keyboard.</td>
</tr>
</tbody>
</table>
## Command syntax formats

The following table provides the command syntax formats.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Examples</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>monospace font</td>
<td>abcd</td>
<td>User input including keypad symbols, numerals, and alphabetic characters as shown; alphabetic characters are not case sensitive.</td>
</tr>
<tr>
<td>&lt; &gt;</td>
<td>&lt;model name&gt;</td>
<td>A required item in a command line. For example, <code>&lt;model name&gt;</code> in a command line means that the model name parameter is required.</td>
</tr>
<tr>
<td>&lt; &gt;*</td>
<td>&lt;value&gt;*</td>
<td>The asterisk indicates that the item shown in italics must occur one or more times in the command line.</td>
</tr>
<tr>
<td>[ ]</td>
<td>[AC]</td>
<td>Optional item.</td>
</tr>
<tr>
<td>[ ]*</td>
<td>[value]*</td>
<td>The asterisk indicates that there is zero or more occurrences of the specified subject.</td>
</tr>
<tr>
<td>&lt;</td>
<td>&gt;</td>
<td>&lt;YES</td>
</tr>
<tr>
<td>[</td>
<td>]</td>
<td>[ON</td>
</tr>
</tbody>
</table>
Numeric value conventions

The numeric value and expression conventions in the following table not only apply to the PSpice Commands, but also to the device declarations and interactive numeric entries described in subsequent chapters.

Literal numeric values are written in standard floating point notation. PSpice applies the default units for the numbers describing the component values and electrical quantities. However, these values can be scaled by following the number using the appropriate scale suffix as shown in the following table.

<table>
<thead>
<tr>
<th>Scale</th>
<th>Symbol</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-15}$</td>
<td>F</td>
<td>femto-</td>
</tr>
<tr>
<td>$10^{-12}$</td>
<td>P</td>
<td>pico-</td>
</tr>
<tr>
<td>$10^{-9}$</td>
<td>N</td>
<td>nano-</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>U</td>
<td>micro-</td>
</tr>
<tr>
<td>$25.4 \times 10^{-6}$</td>
<td>MIL</td>
<td>--</td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>M</td>
<td>milli-</td>
</tr>
<tr>
<td>$10^{+3}$</td>
<td>K</td>
<td>kilo-</td>
</tr>
<tr>
<td>$10^{+6}$</td>
<td>MEG</td>
<td>mega-</td>
</tr>
<tr>
<td>$10^{+9}$</td>
<td>G</td>
<td>giga-</td>
</tr>
<tr>
<td>$10^{+12}$</td>
<td>T</td>
<td>tera-</td>
</tr>
</tbody>
</table>

* Clock cycle varies and must be set where applicable.
### Numeric expression conventions

Numeric values can also be indirectly represented by parameters; see the \texttt{PARAM} (parameter) command. Numeric values and parameters can be used together to form arithmetic expressions. PSpice expressions can incorporate the intrinsic functions shown in the following table.

The Function column lists expressions that PSpice and PSpice A/D recognize. The Meaning column lists the mathematical definition of the function. There are also some differences between the intrinsic functions available for simulation and those available for waveform analysis. Refer to your PSpice user’s guide for more information about waveform analysis.

<table>
<thead>
<tr>
<th>Function</th>
<th>Meaning</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS(x)</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>ACOS(x)</td>
<td>arccosine of x</td>
<td>-1.0 &lt;= x &lt;= +1.0</td>
</tr>
<tr>
<td>ARCTAN(x)</td>
<td>(\tan^{-1}(x))</td>
<td>result in radians</td>
</tr>
<tr>
<td>ASIN(x)</td>
<td>arcsine of x</td>
<td>-1.0 &lt;= x &lt;= +1.0</td>
</tr>
<tr>
<td>ATAN(x)</td>
<td>(\tan^{-1}(x))</td>
<td>result in radians</td>
</tr>
<tr>
<td>ATAN2(y,x)</td>
<td>arctan of ((y/x))</td>
<td>result in radians</td>
</tr>
<tr>
<td>COS(x)</td>
<td>(\cos(x))</td>
<td>(x) in radians</td>
</tr>
<tr>
<td>COSH(x)</td>
<td>hyperbolic cosine of (x)</td>
<td>(x) in radians</td>
</tr>
<tr>
<td>DDT(x)</td>
<td>time derivative of (x)</td>
<td>transient analysis only</td>
</tr>
<tr>
<td>EXP(x)</td>
<td>(e^{x})</td>
<td></td>
</tr>
<tr>
<td>IF(t, x, y)</td>
<td>x if (t=\text{TRUE})</td>
<td>(t) is a Boolean expression that evaluates to \text{TRUE} or \text{FALSE} and can include logical and relational operators (see \texttt{Command line options for OrCAD applications}). (X) and (Y) are either numeric values or expressions. For example, {IF ( (v(1)&lt;\text{THL}), (v(1)), (v(1)*v(1)/\text{THL}) )}</td>
</tr>
<tr>
<td>IMG(x)</td>
<td>imaginary part of (x)</td>
<td>returns 0.0 for real numbers</td>
</tr>
<tr>
<td>LIMIT(x,min,max)</td>
<td>result is (\text{min}) if (x &lt; \text{min}), (\text{max}) if (x &gt; \text{max}), and (x) otherwise</td>
<td></td>
</tr>
<tr>
<td>LOG(x)</td>
<td>(\ln(x))</td>
<td>log base (e)</td>
</tr>
<tr>
<td>LOG10(x)</td>
<td>(\log(x))</td>
<td>log base 10</td>
</tr>
<tr>
<td>M(x)</td>
<td>magnitude of (x)</td>
<td>this produces the same result as \text{ABS}(x)</td>
</tr>
<tr>
<td>MAX(x,y)</td>
<td>maximum of (x) and (y)</td>
<td></td>
</tr>
<tr>
<td>MIN(x,y)</td>
<td>minimum of (x) and (y)</td>
<td></td>
</tr>
<tr>
<td>P(x)</td>
<td>phase of (x)</td>
<td>returns 0.0 for real numbers</td>
</tr>
</tbody>
</table>
**How to Use This Online Manual**

**Overview**

<table>
<thead>
<tr>
<th>Function</th>
<th>Meaning</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR(x,y)</td>
<td>$</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>or, ${x^{**y}}$</td>
<td></td>
</tr>
<tr>
<td>PWRS(x,y)</td>
<td>$+</td>
<td>x</td>
</tr>
<tr>
<td>R(x)</td>
<td>real part of $x$</td>
<td></td>
</tr>
<tr>
<td>SDT(x)</td>
<td>time integral of $x$</td>
<td>transient analysis only</td>
</tr>
<tr>
<td>SGN(x)</td>
<td>signum function</td>
<td></td>
</tr>
<tr>
<td>SIN(x)</td>
<td>$\sin(x)$</td>
<td>$x$ in radians</td>
</tr>
<tr>
<td>SINH(x)</td>
<td>hyperbolic sine of $x$</td>
<td>$x$ in radians</td>
</tr>
<tr>
<td>STP(x)</td>
<td>1 if $x&gt;0.0$, 0 if $x&lt;0.0$</td>
<td>The unit step function can be used to suppress a value until a given amount of time has passed. For instance, ${v(1)\ast STP(TIME-10ns)}$ gives a value of 0.0 until 10ns has elapsed, then gives $v(1)$.</td>
</tr>
<tr>
<td>SQRT(x)</td>
<td>$x^{1/2}$</td>
<td></td>
</tr>
<tr>
<td>TAN(x)</td>
<td>$\tan(x)$</td>
<td>$x$ in radians</td>
</tr>
<tr>
<td>TANH(x)</td>
<td>hyperbolic tangent of $x$</td>
<td>$x$ in radians</td>
</tr>
<tr>
<td>TABLE</td>
<td>$(x,x_1,y_1,x_2,y_2,...,x_n,y_n)$</td>
<td>Result is the $y$ value corresponding to $x$, when all of the $x_n,y_n$ points are plotted and connected by straight lines. If $x$ is greater than the max $x_n$, then the value is the $y_n$ associated with the largest $x_n$. If $x$ is less than the smallest $x_n$, then the value is the $y_n$ associated with the smallest $x_n$.</td>
</tr>
</tbody>
</table>

* Most numeric specifications in PSpice allow for arithmetic expressions. Some exceptions do exist and are summarized in your PSpice user’s guide. There are also some differences between the intrinsic functions available for simulation and those available for waveform analysis. Refer to your PSpice user’s guide for more information about waveform analysis.
Expressions can contain the standard operators as shown in the following table.

<table>
<thead>
<tr>
<th>Operators</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>arithmetic</strong></td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>addition (or string concatenation)</td>
</tr>
<tr>
<td>-</td>
<td>subtraction</td>
</tr>
<tr>
<td>*</td>
<td>multiplication</td>
</tr>
<tr>
<td>/</td>
<td>division</td>
</tr>
<tr>
<td>**</td>
<td>exponentiation</td>
</tr>
<tr>
<td><strong>logical</strong></td>
<td></td>
</tr>
<tr>
<td>~</td>
<td>unary NOT</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>^</td>
<td>boolean XOR</td>
</tr>
<tr>
<td>&amp;</td>
<td>boolean AND</td>
</tr>
<tr>
<td><strong>relational (within IF( ) functions)</strong></td>
<td></td>
</tr>
<tr>
<td>==</td>
<td>equality test</td>
</tr>
<tr>
<td>!=</td>
<td>non-equality test</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than test</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal to test</td>
</tr>
<tr>
<td>&lt;</td>
<td>less than test</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal to test</td>
</tr>
</tbody>
</table>
Command line options for OrCAD applications

Command files

A command file is an ASCII text file which contains a list of commands to be executed. A command file can be specified in multiple ways:

- at the command line when starting PSpice, Stimulus Editor, or the Model Editor,
- by choosing Run Commands from the File menu and entering a command file name (for PSpice and Stimulus Editor only), or
- at the PROBECMD or STMEDCMD command line, found in the configuration file pspice.ini.

The command file is read by the program and all of the commands contained within the file are performed. When the end of the command file is reached, commands are taken from the keyboard and the mouse. If no command file is specified, all of the commands are received from the keyboard and mouse.

The ability to record a set of commands can be useful when using PSpice, the Model Editor, and Stimulus Editor. This is especially useful in PSpice, if you are repeatedly doing the same simulation and looking at the same waveform with only slight changes to the circuit before each run. It can also be used to automatically create hardcopy output at the end of very long (such as overnight) simulation runs.

Creating and editing command files

You can create your own command file using a text editor (such as Notepad). In PSpice and Stimulus Editor, you can choose Log Commands from the File menu (see Log files for an example) to record a list of transactions in a log file, then choose Run Commands from the File menu to run the logged file.

After you activate cursors (from the Tools menu, choose Cursor), any mouse or keyboard movements that you make for moving the cursor will not be recorded in the command file.

If you choose to create a command file using a text editor, note that the commands in the command file are the same as those available from the keyboard with these differences:

- The name of the command or its first capitalized letter can be used.
- Any line that begins with an * is a comment.
- Blank lines are ignored, therefore, they can be added to improve the readability of the command file.
- The commands @CR, @UP, @DWN, @LEFT, @RIGHT, and @ESC are used to represent the <Enter>, <↑>, <↓>, <←>, <→>, and <Esc> keys, respectively.
• The command PAUSE causes PSpice, the Model Editor, or Stimulus Editor to wait until any key on the keyboard is pressed. In the case of PSpice, this can be useful to examine a waveform before the command file draws the next one.

The commands are one to a line in the file, but comment and blank lines can be used to make the file easier to read.

Assuming that a waveform data file has been created by simulating the circuit example.dsn, you can manually create a command file (using a text editor) called example.cmd which contains the commands listed below. This set of commands draws a waveform, allows you to look at it, and then exits PSpice.

* Display trace v(out2) and wait
Trace Add
v(out2)
Pause
* Exit Probe environment
File Exit

See Simulation command line specification format and Specifying simulation command line options for specifying command files on the simulation command line. See Simulation command line specification format and Specifying simulation command line options for details on specifying the /C or -c option for PSpice.

Log files

Instead of creating command files by hand, using a text editor, you can generate them automatically by creating a log file while running PSpice, the Model Editor, or Stimulus Editor. While executing the particular package, all of the commands given are saved in the log file. The format of the log file is correct for use as a command file.

To create a .log file in PSpice or Stimulus Editor, from the File menu, choose Log Commands and enter a log file name. This turns logging on. Any action taken after starting Log Commands is logged in the named file and can be run in another session by choosing Run Commands.

You can also create a log file for PSpice, Stimulus Editor, or the Model Editor by using the /l or -l option at the command line. For example:

PROBE /L EXAMPLE.LOG

Of course, you can use a name for the log file that is more recognizable, such as acplots.cmd (to PSpice, the Model Editor, and Stimulus Editor, the file name is any valid file name for your computer).

You can use either (/) or (-) as separators, and file names can be in upper or lower case.
Editing log files

After PSpice, the Model Editor, or Stimulus Editor is finished, the log file is available for editing to customize it for use as a command file. You can edit the following items:

- Add blank lines and comments to improve readability (perhaps a title and short discussion of what the file does).
- Add the Pause command for viewing waveforms before proceeding.
- Remove the Exit command from the end of the file, so that PSpice, the Model Editor, and Stimulus Editor do not automatically exit when the end of the command file is reached.

You can add or delete other commands from the file or even change the file name to be more recognizable. It is possible to build onto log files, either by using your text editor to combine files or by running PSpice, the Model Editor, and Stimulus Editor with both a command and log file:

`PROBE /C IN.CMD /L OUT.LOG`

The file `in.cmd` gives the command to PSpice, and PSpice saves the (same) commands into the `out.log` file. When `in.cmd` runs out of commands, and PSpice is taking commands from the keyboard, these commands also go into the `out.log` file.

To log commands in PSpice

Use command logging in PSpice to record and save frequently used actions to a command file. Command files are useful when you need to remember the steps taken in order to display a set of waveforms for any given data file.

1. From the File menu, choose Log Commands.
2. In the Log File Name text box, type `2traces`, then click OK.
   
   A check mark appears next to Log Command to indicate that logging is turned on.
3. From the File menu, choose Open.
4. Select `example.dat` (located in the examples directory), then click OK.
5. From the Trace menu, choose Add.
6. Select V(OUT1) and V(OUT2), then click OK.
7. From the File menu, choose Log Commands to turn command logging off.
   
   The check mark next to the command disappears. Subsequent actions performed are not logged in the command file.

You can view the command file using an ASCII text editor, such as Notepad. Command files can be edited or appended, depending on the types of commands you want to store for future use. The file `2traces.cmd` should look as shown below (with the exception of a different file path).

```
*Command file created by Probe - Wed Apr 17 10:33:55
File Open
/orcad/probe/example.dat
OK
Trace Add
V(OUT1) V(OUT2)
OK
```
To run the command log

1. From the File menu, choose Run Command.
2. Select 2traces.cmd, then click OK. The two traces appear.

Simulation command line specification format

The format for specifying command line options for PSpice and PSpice A/D are as follows.
pspice [options] [input file(s)]

input file
Specifies the name of a circuit file for PSpice or PSpice A/D to simulate after it starts. The input file can be a simulation file (.sim, .cir, .net), data files (.dat), output files (.out), or any files (*.*). PSpice opens any files whose extension PSpice does not recognize as a text file.

You can specify multiple input files, but if the output file or data file options are specified, they apply only to the first specified input file.

The input file name can include wildcard characters (* and ?), in which case all file names matching the specification are simulated.

options
One or more of the options listed in Simulation command line options.
## Simulation command line options

Options can be entered using the dash (-) or slash (/) separator.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-bf&lt;flush interval&gt;</td>
<td>Determines how often (in minutes) the simulator will flush the buffers of the waveform data file to disk. This is useful when a long simulation is left running and the machine crashes or is restarted. In this case, the data file will be readable up to the last flush. The default is to flush every 10 minutes. The &lt;flush interval&gt; can be set between 0 and 1440 minutes. A value of zero means not to write unless necessary.</td>
</tr>
<tr>
<td>-bn&lt;number of buffers&gt;</td>
<td>Determines the number of buffers to potentially allocate for the waveform data file. Zero buffers means to do all writing directly to disk. Allocating a large number of buffers can speed up a large simulation, but will increase memory requirements. Exceeding physical memory will either slow down the simulation, or will make it fail. The default number of buffers is 4 (1 buffer if you are using the CSDF option).</td>
</tr>
<tr>
<td>-bs&lt;buffer size factor&gt;</td>
<td>Determines the size of the individual buffers for writing the waveform data file. Using a larger buffer size can reduce execution time, but at the expense of increasing the memory requirements. The values for the buffer files work as follows: option:-bs0 -bs1 -bs2 -bs3 -bs4 -bs5 -bs6 value: 256 512 1024 2048 4096 8192 16384 The default is 4K (8K if you are using CSDF).</td>
</tr>
<tr>
<td>-@ &lt;command file&gt;</td>
<td>Specifies the name of the command file to run.</td>
</tr>
<tr>
<td>-c &lt;file name&gt;</td>
<td>Specifies the command file, which runs the session until the command file ends or PSpice stops.</td>
</tr>
<tr>
<td>-d &lt;data file&gt;</td>
<td>Specifies the name of the waveform data file to which PSpice saves the waveform data from the simulation. By default, the name of the waveform data file is the name of the input file with a .dat extension.</td>
</tr>
<tr>
<td>-e</td>
<td>Exits PSpice after all specified files have been simulated. This option replaces the -wONLY option.</td>
</tr>
<tr>
<td>-i &lt;ini file name&gt;</td>
<td>Specifies the name of an alternate initialization file. If not specified, the simulator uses: \windows\pspice.ini</td>
</tr>
<tr>
<td>-l &lt;file name&gt;</td>
<td>Creates a log file, which saves the commands from this session. This log file can later be used as an input command file for PSpice.</td>
</tr>
<tr>
<td>-o &lt;output file&gt;</td>
<td>Specifies the output file to which PSpice saves the simulation output. By default, the name of the output file name defaults to the name of the input file with an .out extension.</td>
</tr>
</tbody>
</table>
How to Use This Online Manual

Command line options for OrCAD applications

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-p &lt;file name&gt;</td>
<td>Specifies a file that contains goal functions for Performance Analysis, macro definitions, and display configurations. This file is loaded after the global .prb file (specified in the .ini file by the line PRBFILE=pspice.prb), and the local .prb file (&lt;file name&gt;.prb), have been loaded. Definitions in this file will replace definitions from the global or local .prb files that have already been loaded.</td>
</tr>
<tr>
<td>-r</td>
<td>Runs simulation files. If this option is not specified, the specified files are opened but not simulated.</td>
</tr>
<tr>
<td>-t &lt;temp directory name&gt;</td>
<td>Specifies a directory where PSpice can write temporary files. This option replaces the -wTEMP option.</td>
</tr>
<tr>
<td>-wOUT=&lt;suffix&gt;</td>
<td>Specifies the file suffix for the simulation output file. If &lt;suffix&gt; is not specified, the default .out file is used.</td>
</tr>
<tr>
<td>-wDAT=&lt;suffix&gt;</td>
<td>Specifies the file suffix for the waveform data file. If &lt;suffix&gt; is not specified, the default .dat file is used.</td>
</tr>
<tr>
<td>-wTXT=&lt;suffix&gt;</td>
<td>Specifies the file suffix for the CSDF file. If &lt;suffix&gt; is not specified, the default .txt file is used.</td>
</tr>
<tr>
<td>-wNO_NOTIFY</td>
<td>Indicates that the simulator should not display the status message dialog after completion of each circuit file.</td>
</tr>
<tr>
<td>-wPAUSE=&lt;seconds&gt;</td>
<td>Specifies the maximum time that the status dialog box should be displayed. If &lt;seconds&gt; elapses before you click one of the buttons, the dialog will close.</td>
</tr>
</tbody>
</table>

Specifying simulation command line options

Using the pspice.ini configuration file

You can customize your initialization file to include command line options by editing the PSPICECMDLINE line in the file pspice.ini, using any ASCII text editor, such as Notepad. These options take effect the next time PSpice A/D starts.

PSpice options can also be specified on the PSpice command line by typing the executable names.

The command line options can be separated by spaces or in a continuous string, therefore:

- c makeplot.cmd - p newamp.prb
- c makeplot.cmd - p newamp.prb

are equivalent. The order of the options does not matter.

The command line options that use <file name> assume default extensions. These command line options can be used without specifying the extension to <file name>. For example:

- c makeplot - p newamp
- c makeplot.cmd - p newamp.prb

are equivalent. However, PSpice searches first for the exact <file name> specified for these command line options, and if that <file name> exists, PSpice uses it. If the exact <file name>
If the specified file does not exist, PSpice adds default extensions to `<file name>` and searches for those. The following default extensions are used:

<table>
<thead>
<tr>
<th>Extension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;file name.dat&gt;</code></td>
<td>waveform data file</td>
</tr>
<tr>
<td><code>-c&lt;file name.cmd&gt;</code></td>
<td>command file</td>
</tr>
<tr>
<td><code>-l&lt;file name.log&gt;</code></td>
<td>log file</td>
</tr>
<tr>
<td><code>-p&lt;file name.prb&gt;</code></td>
<td>displays, goal functions, and macros file</td>
</tr>
</tbody>
</table>

You can learn more about PSpice macros by consulting PSpice Help.
### Commands

#### standard analyses
- `.AC` (AC analysis)
- `.DC` (DC analysis)
- `.FOUR` (Fourier analysis)
- `.NOISE` (noise analysis)
- `.OP` (bias point)
- `.SENS` (sensitivity analysis)
- `.TF` (transfer)
- `.TRAN` (transient analysis)

#### output control
- `.PLOT` (plot)
- `.PRINT` (print)
- `.PROBE` (Probe)
- `.VECTOR` (digital output)
- `.WATCH` (watch analysis results)

#### simple multi-run analyses
- `.STEP` (parametric analysis)
- `.TEMP` (temperature)

#### circuit file processing
- `.END` (end of circuit)
- `.FUNC` (function)
- `.INC` (include file)
- `.LIB` (library file)
- `.PARAM` (parameter)

#### statistical analyses
- `.MC` (Monte Carlo analysis)
- `.WCASE` (sensitivity/ worst-case analysis)

#### device modeling
- `.ENDS` (end subcircuit)
- `.DISTRIBUTION` (user-defined distribution)
- `.MODEL` (model definition)
- `.SUBCKT` (subcircuit)

#### initial conditions
- `.IC` (initial bias point condition)
- `.LOADBIAS` (load bias point file)
- `.NODESET` (set approximate node voltage for bias point)
- `.SAVEBIAS` (save bias point to file)

#### miscellaneous
- `.ALIASES, .ENDALIASES` (aliases and endaliases)
- `.EXTERNAL` (external port)
- `.OPTIONS` (analysis options)
- `.STIMLIB` (stimulus library file)
- `.STIMULUS` (stimulus)
- `.TEXT` (text parameter)
- `*` (comment)
- `;` (in-line comment)
- `+` (line continuation)
Schematics users enter analysis specifications through the Analysis Setup dialog box (from the Analysis menu, select Setup).

<table>
<thead>
<tr>
<th>Function</th>
<th>PSpice command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>standard analyses</td>
<td>.AC (AC analysis)</td>
<td>frequency response</td>
</tr>
<tr>
<td></td>
<td>.DC (DC analysis)</td>
<td>DC sweep</td>
</tr>
<tr>
<td></td>
<td>.FOUR (Fourier analysis)</td>
<td>Fourier components</td>
</tr>
<tr>
<td></td>
<td>.NOISE (noise analysis)</td>
<td>noise</td>
</tr>
<tr>
<td></td>
<td>.OP (bias point)</td>
<td>bias point</td>
</tr>
<tr>
<td></td>
<td>.SENS (sensitivity analysis)</td>
<td>DC sensitivity</td>
</tr>
<tr>
<td></td>
<td>.TF (transfer)</td>
<td>small-signal DC transfer function</td>
</tr>
<tr>
<td></td>
<td>.TRAN (transient analysis)</td>
<td>transient</td>
</tr>
<tr>
<td>simple multi-run analyses</td>
<td>.STEP (parametric analysis)</td>
<td>parametric</td>
</tr>
<tr>
<td></td>
<td>.TEMP (temperature)</td>
<td>temperature</td>
</tr>
<tr>
<td>statistical analyses</td>
<td>.MC (Monte Carlo analysis)</td>
<td>Monte Carlo</td>
</tr>
<tr>
<td></td>
<td>.WCASE (sensitivity/worst-case analysis)</td>
<td>sensitivity/worst-case</td>
</tr>
<tr>
<td>initial conditions</td>
<td>.IC (initial bias point condition)</td>
<td>clamp node voltage for bias point calculation</td>
</tr>
<tr>
<td></td>
<td>.LOADBIAS (load bias point file)</td>
<td>to restore a .NODESET bias point</td>
</tr>
<tr>
<td></td>
<td>.NODESET (set approximate node voltage for bias point)</td>
<td>to suggest a node voltage for bias calculation</td>
</tr>
<tr>
<td></td>
<td>.SAVEBIAS (save bias point to file)</td>
<td>to store .NODESET bias point information</td>
</tr>
<tr>
<td>device modeling</td>
<td>.ENDS (end subcircuit)</td>
<td>end of subcircuit definition</td>
</tr>
<tr>
<td></td>
<td>.DISTRIBUTION (user-defined distribution)</td>
<td>model parameter tolerance distribution</td>
</tr>
<tr>
<td></td>
<td>.MODEL (model definition)</td>
<td>modeled device definition</td>
</tr>
<tr>
<td></td>
<td>.SUBCKT (subcircuit)</td>
<td>to start subcircuit definition</td>
</tr>
<tr>
<td>output control</td>
<td>.PLOT (plot)</td>
<td>to send an analysis plot to output file</td>
</tr>
<tr>
<td></td>
<td>.PRINT (print)</td>
<td>(line printer format)</td>
</tr>
<tr>
<td></td>
<td>.PROBE (Prober)</td>
<td>to send an analysis table to output file</td>
</tr>
<tr>
<td></td>
<td>.VECTOR (digital output)</td>
<td>to send simulation results to Probe data file</td>
</tr>
<tr>
<td></td>
<td>.WATCH (watch analysis results)</td>
<td>digital state output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>view numerical simulation results in progress</td>
</tr>
<tr>
<td>Function</td>
<td>PSpice command</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>circuit file</td>
<td>.END (end of circuit)</td>
<td>end of circuit simulation description</td>
</tr>
<tr>
<td>processing</td>
<td>.FUNC (function)</td>
<td>expression function definition</td>
</tr>
<tr>
<td></td>
<td>.INC (include file)</td>
<td>include specified file</td>
</tr>
<tr>
<td></td>
<td>.LIB (library file)</td>
<td>reference specified library</td>
</tr>
<tr>
<td></td>
<td>.PARAM (parameter)</td>
<td>parameter definition</td>
</tr>
<tr>
<td>miscellaneous</td>
<td>.ALIASES, .ENDALIASES (aliases</td>
<td>to begin and end an alias definition</td>
</tr>
<tr>
<td></td>
<td>and endaliases)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>.EXTERNAL (external port)</td>
<td>to identify nets representing the outermost (or peripheral) connections to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the circuit being simulated</td>
</tr>
<tr>
<td></td>
<td>.OPTIONS (analysis options)</td>
<td>to set miscellaneous simulation limits, analysis control parameters, and output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>characters</td>
</tr>
<tr>
<td></td>
<td>.STIMLIB (stimulus library file)</td>
<td>to specify a stimulus library name containing .STIMULUS information</td>
</tr>
<tr>
<td></td>
<td>.STIMULUS (stimulus)</td>
<td>stimulus device definition</td>
</tr>
<tr>
<td></td>
<td>.TEXT (text parameter)</td>
<td>text expression, parameter, or file name used by digital devices</td>
</tr>
<tr>
<td></td>
<td>* (comment)</td>
<td>to create a comment line</td>
</tr>
<tr>
<td></td>
<td>; (in-line comment)</td>
<td>to add an in-line comment</td>
</tr>
<tr>
<td></td>
<td>+ (line continuation)</td>
<td>to continue the text of the previous line</td>
</tr>
</tbody>
</table>
.AC (AC analysis)

Purpose
The .AC command calculates the frequency response of a circuit over a range of frequencies.

General form
.AC <sweep type> <points value>
+ <start frequency value> <end frequency value>

Examples
.AC LIN 10 10Hz 200Hz
.AC OCT 1 1kHz 16kHz
.AC DEC 2 1MEG 100MEG

Arguments and options
<sweep type>
Must be LIN, OCT, or DEC, as described below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIN</td>
<td>linear sweep</td>
<td>The frequency is swept linearly from the starting to the ending frequency. The &lt;points value&gt; is the total number of points in the sweep.</td>
</tr>
<tr>
<td>OCT</td>
<td>sweep by octaves</td>
<td>The frequency is swept logarithmically by octaves. The &lt;points value&gt; is the number of points per octave.</td>
</tr>
<tr>
<td>DEC</td>
<td>sweep by decades</td>
<td>The frequency is swept logarithmically by decades. The &lt;points value&gt; is the number of points per decade.</td>
</tr>
</tbody>
</table>

<points value>
Specifies the number of points in the sweep, using an integer.

<start frequency value> <end frequency value>
The end frequency value must not be less than the start frequency value, and both must be greater than zero. The whole sweep must include at least one point. If a group delay (G suffix) is specified as an output, the frequency steps must be close enough together that the phase of that output changes smoothly from one frequency to the next. Calculate group delay by subtracting the phases of successive outputs and dividing by the frequency increment.

Comments
A .PRINT (print), .PLOT (plot), or .PROBE (Probe) command must be used to get the results of the AC sweep analysis.

AC analysis is a linear analysis. The simulator calculates the frequency response by linearizing the circuit around the bias point.

All independent voltage and current sources that have AC values are inputs to the circuit. During AC analysis, the only independent sources that have nonzero amplitudes are those using AC specifications. The SIN specification does not count, as it is used only during transient analysis.

To analyze nonlinear functions such as mixers, frequency doublers, and AGC, use .TRAN (transient analysis).
.ALIASES, .ENDALIASES (aliases and endaliases)

Purpose
The Alias commands set up equivalences between node names and pin names, so that traces in the Probe display can be identified by naming a device and pin instead of a node. They are also used to associate a net name with a node name.

General form
```
.ALIIASES
  <device name>  <device alias> (<<(pin)>>=<<(node)>>)
  (<<(net)>>=<<(node)>>)
.ENDALIASES
```

Examples
```
.ALIIASES
R_RBIAS  RBIAS (1=$N_0001 2=VDD)
Q_Q3  Q3 (c=$N_0001 b=$N_0001 e=VEE)
  (OUT=$N_0007)
.ENDALIASES
```

The first alias definition shown in the example allows the name RBIAS to be used as an alias for R_RBIAS, and it relates pin 1 of device R_RBIAS to node $N_0001 and pin 2 to VDD.

The last alias definition equates net name OUT to node name $N_0007.
.DC (DC analysis)

Purpose
The .DC command performs a linear, logarithmic, or nested DC sweep analysis on the circuit. The DC sweep analysis calculates the circuit’s bias point over a range of values for <sweep variable name>.

Sweep type
The sweep can be linear, logarithmic, or a list of values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIN</td>
<td>linear sweep</td>
<td>The sweep variable is swept linearly from the starting to the ending value.</td>
</tr>
<tr>
<td>OCT</td>
<td>sweep by octaves</td>
<td>Sweep by octaves. The sweep variable is swept logarithmically by octaves.</td>
</tr>
<tr>
<td>DEC</td>
<td>sweep by decades</td>
<td>Sweep by decades. The sweep variable is swept logarithmically by decades.</td>
</tr>
<tr>
<td>LIST</td>
<td>list of values</td>
<td>Use a list of values.</td>
</tr>
</tbody>
</table>
Linear sweep

General form

```
.DC [LIN] <sweep variable name>
+ <start value> <end value> <increment value>
+ [nested sweep specification]
```

Examples

```
.DC VIN -.25 .25 .05
.DC LIN I2 5mA -2mA 0.1mA
.DC VCE OV 10V .5V 1B 0mA 1mA 50uA
.DC RES RMOD(R) 0.9 1.1 .001
```

Arguments and options

- `<start value>`
  - Can be greater or less than `<end value>`: that is, the sweep can go in either direction.

- `<increment value>`
  - The step size. This value must be greater than zero.

Comments

- The sweep variable is swept linearly from the starting to the ending value.
- The keyword LIN is optional.

Logarithmic sweep

General form

```
.DC <logarithmic sweep type> <sweep variable name>
+ <start value> <end value> <points value>
+ [nested sweep specification]
```

Examples

```
.DC DEC NPN QFAST(IS) 1E-18 1E-14 5
```

Arguments and options

- `<logarithmic sweep type>`
  - Must be specified as either DEC (to sweep by decades) or OCT (to sweep by octaves).

- `<start value>`
  - Must be positive and less than `<end value>`.

- `<points value>`
  - The number of steps per octave or per decade in the sweep. This value must be an integer.

Comments

- Either OCT or DEC must be specified for the `<logarithmic sweep type>`.
Nested sweep

General form

```
.DC <sweep variable name> LIST <value>*
+ [nested sweep specification]
```

Examples

```
.DC TEMP LIST 0 20 27 50 80 100 PARAM Vsupply 7.5 15 .5
```

Arguments and options

<sweep variable name>
After the DC sweep is finished, the value associated with <sweep variable name> is set back to the value it had before the sweep started. The following items can be used as sweep variables in a DC sweep:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>A name of an independent voltage or current source.</td>
<td>During the sweep, the source’s voltage or current is set to the sweep value.</td>
</tr>
<tr>
<td>Model Parameter</td>
<td>A model type and model name followed by a model parameter name in parenthesis.</td>
<td>The parameter in the model is set to the sweep value. The following model parameters cannot be (usefully) swept: L and W for the MOSFET device (use LD and WD as a work around), and any temperature parameters, such as TC1 and TC2 for the resistor.</td>
</tr>
<tr>
<td>Temperature</td>
<td>Use the keyword TEMP for &lt;sweep variable name&gt;.</td>
<td>Set the temperature to the sweep value. For each value in the sweep, all the circuit components have their model parameters updated to that temperature.</td>
</tr>
<tr>
<td>Global Parameter</td>
<td>Use the keyword PARAM, followed by the parameter name, for &lt;sweep variable name&gt;.</td>
<td>During the sweep, the global parameter’s value is set to the sweep value and all expressions are reevaluated.</td>
</tr>
</tbody>
</table>

Comments

For a nested sweep, a second sweep variable, sweep type, start, end, and increment values can be placed after the first sweep. In the nested sweep example, the first sweep is the inner loop: the entire first sweep is performed for each value of the second sweep.

When using a list of values, there are no start and end values. Instead, the numbers that follow the keyword LIST are the values that the sweep variable is set to.

The rules for the values in the second sweep are the same as for the first. The second sweep generates an entire PRINT (print) table or PLOT (plot) plot for each value of the sweep. Probe displays nested sweeps as a family of curves.
The .DISTRIBUTION command defines a user distribution for tolerances, and is only used with Monte Carlo and sensitivity/worst-case analyses. The curve described by a .DISTRIBUTION command controls the relative probability distribution of random numbers generated by PSpice to calculate model parameter deviations.

### General form

```
DISTRIBUTION <name> (<deviation> <probability>)*
```

### Examples

```
.DISTRIBUTION bi_modal (-1.1) (-.5.1) (.5.0) (.5.0) + (.5,1) (1,1)
.DISTRIBUTION triangular (-1,0) (0,1) (1,0)
```

### Arguments and options

```
(<deviation> <probability>)
```

- **Defines the distribution curve by pairs, or corner points, in a piecewise linear fashion. You can specify up to 100 value pairs.**

```
<deviation>
```

- **Must be in the range (-1,+1), which matches the range of the random number generator. No <deviation> can be less than the previous <deviation> in the list, although it can repeat the previous value.**

```
<probability>
```

- **Represents a relative probability, and must be positive or zero.**

### Comments

When using Schematics, several distributions can be defined by configuring an include file containing the .DISTRIBUTION command. For details on how to do this, refer to your PSpice user’s guide.

If you are not using Schematics, a user-defined distribution can be specified as the default by setting the DISTRIBUTION parameter in the .OPTIONS (analysis options) command.

### Deriving updated parameter values

The updated value of a parameter is derived from a combination of a random number, the distribution, and the tolerance specified. This method permits distributions which have different excursions in the positive and negative directions. It also allows the use of one distribution even if the tolerances of the components are different so long as the general shape of the distributions are the same.

1. Generate a <temporary random number> in the range (0, 1).
2. Normalize the area under the specified distribution.
3. Set the <final random number> to the point where the area under the normalized distribution equals the <temporary random number>.
4. Multiply this <final random number> by the specified tolerance.
Usage example

To illustrate, assume there is a 1.0 µfd capacitor that has a variation of -50% to +25%, and another that has tolerances of -10% to +5%. Note that both capacitors’ tolerances are in the same general shape, i.e., both have negative excursions twice as large as their positive excursions.

```
.distribution cdistrib (-1,1) (.5, 1) (.5, 0) (1, 0)
c1 1 0 cmod 11u
c2 1 0 cmod2 1u
.model cmod1 cap (c=1 dev/cdistrib 50%)
.model cmod2 cap (c=1 dev/cdistrib 10%)
```

The steps taken for this example are as follows:

1. Generate a <temporary random value> of 0.3.
2. Normalize the area under the cdistrib distribution (1.5) to 1.0.
3. The <final random number> is therefore -0.55 (the point where the normalized area equals 0.3).
4. For c1, this -0.55 is then scaled by 50%, resulting in -0.275; for c2, it is scaled by 10%, resulting in -0.055.

Separate random numbers are generated for each parameter that has a tolerance unless a tracking number is specified.
The .END command marks the end of the circuit. All the data and every other command must come before it. When the .END command is reached, PSpice does all the specified analyses on the circuit.

General form

.END

Examples

* 1st circuit in file
  ... circuit definition
  .END
* 2nd circuit in file
  ... circuit definition
  .END

Comments

There can be more than one circuit in an input file. Each circuit is marked by an .END command. PSpice processes all the analyses for each circuit before going on to the next one. Everything is reset at the beginning of each circuit. Having several circuits in one file gives the same results as having them in separate files and running each one separately. However, all the simulation results go into one .OUT file and one .DAT file. This is a convenient way to arrange a set of runs for overnight operation.

The last statement in an input file must be an .END command.
Commands

`.EXTERNAL (external port)`

**Purpose**
External ports are provided as a means of identifying and distinguishing those nets representing the outermost (or peripheral) connections to the circuit being simulated. The external port statement `.EXTERNAL` applies only to nodes that have digital devices attached to them.

**General form**
```
.EXTERNAL <attribute> <node-name>*
```

**Examples**
```
.EXTERNAL INPUT Data1, Data2, Data3
.EXTERNAL OUTPUT P1
.EXTERNAL BIDIRECTIONAL BPort1 BPort2 BPort3
```

**Arguments and options**

- `<attribute>`
  One of the keywords INPUT, OUTPUT, or BIDIRECTIONAL, describing the usage of the port.

- `<node_name>`
  One or more valid PSpice A/D node names.

**Comments**

When a node is included in a `.EXTERNAL` statement it is identified as a primary observation point. For example, if you are modeling and simulating a PCB-level description, you could place an `.EXTERNAL` (or its Capture symbol counterparts) on the edge pin nets to describe the pin as the external interface point of the network.

PSpice recognizes the nets marked as `.EXTERNAL` when reporting any sort of timing violation. When a timing violation occurs, PSpice analyzes the conditions that would permit the effects of such a condition to propagate through the circuit. If, during this analysis, a net marked external is encountered, PSpice reports the condition as a Persistent Hazard, signifying that it has a potential effect on the externally visible behavior of the circuit. For more information on Persistent Hazards, refer to your PSpice user’s guide.

Port specifications are inserted in the netlist by Capture whenever an external port symbol, `EXTERNAL_IN`, `EXTERNAL_OUT`, or `EXTERNAL_BI` is used. Refer to your PSpice user’s guide for more information.
.FOUR (Fourier analysis)

Purpose
Fourier analysis decomposes the results of a transient analysis into Fourier components.

General form
.FOUR <frequency value> [no. harmonics value] <output variable>

Examples
.FOUR 10kHz V(5) V(6,7) I(VSENS3)
.FOUR 60Hz 20 V(17)
.FOUR 10kHz V([OUT1],[OUT2])

Arguments and options

<output variable>
An output variable of the same form as in a .PRINT (print) command or .PLOT (plot) command for a transient analysis.

<frequency value>
The fundamental frequency. Not all of the transient results are used, only the interval from the end, back to 1/<frequency value> before the end is used. This means that the transient analysis must be at least 1/<frequency value> seconds long.

Comments
The analysis results are obtained by performing a Fourier integral on the results from a transient analysis. The analysis must be supplied with specified output variables using evenly spaced time points. The time interval used is <print step value> in the .TRAN (transient analysis) command, or 1% of the <final time value> (TSTOP) if smaller, and a 2nd-order polynomial interpolation is used to calculate the output value used in the integration. The DC component, the fundamental, and the 2nd through 9th harmonics of the selected voltages and currents are calculated by default, although more harmonics can be specified.

A .FOUR command requires a .TRAN command, but Fourier analysis does not require .PRINT, .PLOT, or .PROBE (Probe) commands. The tabulated results are written to the output file (.out) as the transient analysis is completed.

The results of the .FOUR command are only available in the output file. They cannot be viewed in Probe.
.FUNC (function)

Purpose

The .FUNC command defines functions used in expressions. Besides their obvious flexibility, they are useful for where there are several similar subexpressions in a circuit file.

General form

.FUNC <name> ([arg]*) {<body>}

Examples

.FUNC E(x) {exp(x)}
.FUNC DECAY(CNST) {E(-CNST*TIME)}
.FUNC TRIWAV(x) {ACOS(COS(x))/3.14159}
.FUNC MIN3(A,B,C) {MIN(A,MIN(B,C))}

Arguments and options

.FUNC

Does not have to precede the first use of the function name. Functions cannot be redefined and the function name must not be the same as any of the predefined functions (e.g., SIN and SQRT). See How to Use This Online Manual for a list of valid expressions. .FUNC arguments cannot be node names.

<body>

Refers to other (previously defined) functions; the second example, DECAY, uses the first example, E.

[arg]

Specifies up to 10 arguments in a definition. The number of arguments in the use of a function must agree with the number in the definition. Functions can be defined as having no arguments, but the parentheses are still required. Parameters, TIME, other functions, and the Laplace variable s are allowed in the body of function definitions.

Comments

The <body> of a defined function is handled in the same way as any math expression; it is enclosed in curly braces {}. Previous versions of PSpice did not require this, so for compatibility the <body> can be read without braces, but a warning is generated.

Creating a file of frequently used .FUNC definitions and accessing them using an .INC command near the beginning of the circuit file can be helpful. .FUNC commands can also be defined in subcircuits. In those cases they only have local scope.
.IC (initial bias point condition)

Purpose
The .IC command sets initial conditions for both small-signal and transient bias points. Initial conditions can be given for some or all of the circuit’s nodes.

.IC sets the initial conditions for the bias point only. It does not affect a .DC (DC analysis) sweep.

General form
.IC < V(<node> [,<node>])=<value> >*
.IC <I(<inductor>)=<value>>*

Examples
.IC V(2)=3.4 V(102)=0 V(3)=-1V I(L1)=2uAmp
.IC V(InPlus,InMinus)=1e-3 V(100,133)=5.0V

Arguments and options

<value>
A voltage assigned to <node> (or a current assigned to an inductor) for the duration of the bias point calculation.

Comments
The voltage between two nodes and the current through an inductor can be specified. During bias calculations, PSpice clamps the voltages to specified values by attaching a voltage source with a 0.0002 ohm series resistor between the specified nodes. After the bias point has been calculated and the transient analysis started, the node is released.

If the circuit contains both the .IC command and .NODESET (set approximate node voltage for bias point) command for the same node or inductor, the .NODESET command is ignored (.IC overrides .NODESET).

Refer to your PSpice user’s guide for more information on setting initial conditions.

An .IC command that imposes nonzero voltages on inductors cannot work properly, since inductors are assumed to be short circuits for bias point calculations. However, inductor currents can be initialized.
.INC (include file)

Purpose
The .INC command inserts the contents of another file.

General form
.INC <file name>

Examples
.INC "SETUP.CIR"
.INC "C:\LIB\VCO.CIR"

Arguments and options

Arguments

<file name>
Any character string that is a valid file name for your computer system.

Comments
Including a file is the same as bringing the file’s text into the circuit file. Everything in the included file is actually read in. The comments of the included file are then treated just as if they were found in the parent file.

Included files can contain any valid PSpice statements, with the following conditions:

• The included files should not contain title lines unless they are commented.
• Included files can be nested up to 4 levels.

Every model and subcircuit definition, even if not needed, takes up memory.
.LIB (library file)

**Purpose**
The .LIB command references a model or subcircuit library in another file.

**General form**
```
.LIB [file_name]
```

**Examples**
- `.LIB linear.lib`
- `.LIB "C:\lib\bipolar.lib"`

**Arguments and options**
- `[file_name]`
  - Any character string that is a valid file name for the computer system.

**Comments**
Library files can contain any combination of the following:
- `.MODEL (model definition)` commands
- .PARAM (parameter) commands
- .FUNC (function) commands
- .LIB commands

No other statements are allowed. For further discussion of library files, refer to your PSpice user's guide.

If `[file_name]` is left off, all references point to the master library file, `nom.lib`. When a library file is referenced in Schematics, PSpice first searches for the file in the current working directory, then searches in the directory specified by the LIBPATH variable (set in `msim.ini`).

When any library is modified, PSpice creates an index file based on the first use of the library. The index file is organized so that PSpice can find a particular .MODEL or .SUBCKT (subcircuit) quickly, despite the size of the library file.

```
The index files have to be regenerated each time the library is changed. Because of this, it is advantageous to configure separately any frequently changed libraries.
```

`Nom.lib` normally contains references to all parts in the MicroSim Standard Model Library. You can edit `nom.lib` to include your custom model references.
.LOADBIAS (load bias point file)

**Purpose**
The .LOADBIAS command loads the contents of a bias point file. It is helpful in setting initial bias conditions for subsequent simulations. However, the use of .LOADBIAS does not guarantee convergence.

**General form**
```
LOADBIAS <file name>
```

**Examples**
```
LOADBIAS "SAVETRAN.NOD"
LOADBIAS "C:\PROJECT\INIT.FIL"
```

**Arguments and options**
```
<file name>
```
Any character string which is a valid computer system file name, but it must be enclosed in quotation marks.

**Comments**
Normally, the bias point file is produced by a previous circuit simulation using the .SAVEBIAS (save bias point to file) command.

The bias point file is a text file that contains one or more comment lines and a .NODESET (set approximate node voltage for bias point) command setting the bias point voltage or inductor current values. If a fixed value for a transient analysis bias point needs to be set, this file can be edited to replace the .NODESET command with an .IC (initial bias point condition) command.

Any nodes mentioned in the loaded file that are not present in the circuit are ignored, and a warning message will be generated.

To echo the .LOADBIAS file contents to the output file, use the EXPAND option on the .OPTIONS (analysis options) command.
.MC (Monte Carlo analysis)

Purpose
The .MC command causes a Monte Carlo (statistical) analysis of the circuit and causes PSpice to perform multiple runs of the selected analysis (DC, AC, or transient).

General form
.MC <#runs value> <analysis> <output variable> <function> [option]*
+ [SEED=value]

Examples
.MC 10 TRAN V(5) YMAX
.MC 50 DC IC(Q7) YMAX LIST
.MC 20 AC VP(13,5) YMAX LIST OUTPUT ALL
.MC 10 TRAN V([OUT1],[OUT2]) YMAX SEED=9321

Arguments and options

<#runs value>
The total number of runs to be performed (for printed results the upper limit is 2,000, and for results to be viewed in Probe, the limit is 400).

<analysis>
Specifies at least one analysis type: .DC (DC analysis), .AC (AC analysis), or .TRAN (transient analysis). This analysis is repeated in subsequent passes. All analyses that the circuit contains are performed during the nominal pass. Only the selected analysis is performed during subsequent passes.

<output variable>
Identical in format to that of a .PRINT (print) output variable.

<function>
Specifies the operation to be performed on the values of <output variable> to reduce these to a single value. This value is the basis for the comparisons between the nominal and subsequent runs. The <function> can be any one of the following:

<table>
<thead>
<tr>
<th>Function</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>YMAX</td>
<td>Find the absolute value of the greatest difference in each waveform from the nominal run.</td>
</tr>
<tr>
<td>MAX</td>
<td>Find the maximum value of each waveform.</td>
</tr>
<tr>
<td>MIN</td>
<td>Find the minimum value of each waveform.</td>
</tr>
<tr>
<td>RISE_EDGE(&lt;value&gt;)</td>
<td>Find the first occurrence of the waveform crossing above the threshold &lt;value&gt;. The waveform must have one or more points at or below &lt;value&gt; followed by one above; the output value listed is the first point that the waveform increases above &lt;value&gt;.</td>
</tr>
<tr>
<td>FALL_EDGE(&lt;value&gt;)</td>
<td>Find the first occurrence of the waveform crossing below the threshold &lt;value&gt;. The waveform must have one or more points at or above &lt;value&gt; followed by one below; the output value listed is where the waveform decreases below &lt;value&gt;.</td>
</tr>
</tbody>
</table>
<function> and all [option]s (except for <output type>) have no effect on the Probe data that is saved from the simulation. They are only applicable to the output file.

[option]*
Can include zero or more of the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST</td>
<td>Lists, at the beginning of each run, the model parameter values actually used for each component during that run.</td>
<td></td>
</tr>
<tr>
<td>OUTPUT</td>
<td>Asks for an output from subsequent runs, after the nominal (first) run. The output from any run is governed by a .PRINT, .PLOT, and .PROBE command in the file. If OUTPUT is omitted, then only the nominal run produces output. The &lt;output type&gt; is one of the ones shown in the examples to the right.</td>
<td>ALL forces all output to be generated (including the nominal run). FIRST &lt;N&gt; generates output only during the first n runs. EVERY &lt;N&gt; generates output every n&lt;sup&gt;th&lt;/sup&gt; run. RUNS &lt;N&gt;* does analysis and generates output only for the listed runs. Up to 25 values can be specified in the list.</td>
</tr>
<tr>
<td>RANGE*</td>
<td>Restricts the range over which &lt;function&gt; is evaluated. An asterisk (*) can be used in place of a &lt;value&gt; to show for all values.</td>
<td>YMAX RANGE(<em>,.5) YMAX is evaluated for values of the sweep variable (e.g., time and frequency) of .5 or less. MAX RANGE(-1,</em>) The maximum of the output variable is found for values of the sweep variable of -1 or more.</td>
</tr>
</tbody>
</table>

* If RANGE is omitted, then <function> is evaluated over the whole sweep range. This is equivalent to RANGE(*,*).

[SEED=value]
Defines the seed for the random number generator within the Monte Carlo analysis (The Art of Computer Programming, Donald Knuth, vol. 2, pg. 171, “subtractive method”).

<value>
Must be an odd integer ranging from 1 to 32,767. If the seed value is not set, its default value is 17,533.

For almost all analyses, the default seed value is adequate to achieve a constant set of results. The seed value can be modified within the integer value as required.
The first run uses nominal values of all components. Subsequent runs use variations on model parameters as specified by the DEV and LOT tolerances on each .MODEL (model definition) parameter.

The other specifications on the .MC command control the output generated by the Monte Carlo analysis.

For more information on Monte Carlo analysis, refer to your PSpice user’s guide.
**.MODEL (model definition)**

**Purpose**
The .MODEL command defines a set of device parameters that can be referenced by devices in the circuit.

**General form**
```
.MODEL <model name> [AKO: <reference model name>]
+ <model type>
+ {{[<parameter name> = <value> [tolerance specification]]}* 
+ [T_MEASURED=<value>] [[T_ABS=<value>] or 
+ [T_REL_GLOBAL=<value>] [T_REL_LOCAL=<value>]]}
```

**Examples**
```
.PROJ
.MODEL RMAX RES (R=1.5 TC1=.02 TC2=.005)
.MODEL DNOM D (IS=1E-9)
.MODEL QDRIV NPN (IS=1E-7 BF=30)
.MODEL MLOAD NMOS(LEVEL=1 VTO=.7 CJ=.02pF)
.MODEL CMOD CAP (C=1 DEV 5%)
.MODEL DLOAD D (IS=1E-9 DEV .5% LOT 10%)
.MODEL RTRACK RES (R=1 DEV/GAUSS 1% LOT/UNIFORM 5%)
.MODEL QDR2 AKO:QDRIV NPN (BF=50 IKF=50m)
```

**Arguments and options**

- `<model name>`
  The model name which is used to reference a particular model.

- `<reference model name>`
  The model types of the current model and the AKO (A Kind Of) reference model must be the same. The value of each parameter of the referenced model is used unless overridden by the current model, e.g., for QDR2 in the last example, the value of IS derives from QDRIV, but the values of BF and IKF come from the current definition. Parameter values or formulas are transferred, but not the tolerance specification. The referenced model can be in the main circuit file, accessed through a .INC command, or it can be in a library file; see .LIB (library file).

- `<model type>`
  Must be one of the types outlined in the table that follows.

  Devices can only reference models of a corresponding type; for example:

  - A JFET can reference a model of types NJF or PJF, but not of type NPN.
  - There can be more than one model of the same type in a circuit, although they must have different names.

  Following the `<model type>` is a list of parameter values enclosed by parentheses. None, any, or all of the parameters can be assigned values. Default values are used for all unassigned parameters. The lists of parameter names, meanings, and default values are found in the individual device descriptions.
<table>
<thead>
<tr>
<th>Model type</th>
<th>Instance name</th>
<th>Type of device</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAP</td>
<td>Cxxx</td>
<td>capacitor</td>
</tr>
<tr>
<td>CORE</td>
<td>Kxxx</td>
<td>nonlinear, magnetic core (transformer)</td>
</tr>
<tr>
<td>D</td>
<td>Dxxx</td>
<td>diode</td>
</tr>
<tr>
<td>DINPUT</td>
<td>Nxxx</td>
<td>digital input device (receive from digital)</td>
</tr>
<tr>
<td>DOUTPUT</td>
<td>Oxxx</td>
<td>digital output device (transmit to digital)</td>
</tr>
<tr>
<td>GASFET</td>
<td>Bxxx</td>
<td>N-channel GaAs MESFET</td>
</tr>
<tr>
<td>IND</td>
<td>Lxxx</td>
<td>inductor</td>
</tr>
<tr>
<td>ISWITCH</td>
<td>Wxxx</td>
<td>current-controlled switch</td>
</tr>
<tr>
<td>LPNP</td>
<td>Qxxx</td>
<td>lateral PNP bipolar transistor</td>
</tr>
<tr>
<td>NIGBT</td>
<td>Zxxx</td>
<td>N-channel insulated gate bipolar transistor (IGBT)</td>
</tr>
<tr>
<td>NJF</td>
<td>Jxxx</td>
<td>N-channel junction FET</td>
</tr>
<tr>
<td>NMOS</td>
<td>Mxxx</td>
<td>N-channel MOSFET</td>
</tr>
<tr>
<td>NPN</td>
<td>Qxxx</td>
<td>NPN bipolar transistor</td>
</tr>
<tr>
<td>PJF</td>
<td>Jxxx</td>
<td>P-channel junction FET</td>
</tr>
<tr>
<td>PMOS</td>
<td>Mxxx</td>
<td>P-channel MOSFET</td>
</tr>
<tr>
<td>PNP</td>
<td>Qxxx</td>
<td>PNP bipolar transistor</td>
</tr>
<tr>
<td>RES</td>
<td>Rxxx</td>
<td>resistor</td>
</tr>
<tr>
<td>TRN</td>
<td>Txxx</td>
<td>lossy transmission line</td>
</tr>
<tr>
<td>UADC</td>
<td>Uxxx</td>
<td>multi-bit analog-to-digital converter</td>
</tr>
<tr>
<td>UDAC</td>
<td>Uxxx</td>
<td>multi-bit digital-to-analog converter</td>
</tr>
<tr>
<td>UDELAY</td>
<td>Uxxx</td>
<td>digital delay line</td>
</tr>
<tr>
<td>UEFF</td>
<td>Uxxx</td>
<td>edge-triggered flip-flop</td>
</tr>
<tr>
<td>UGATE</td>
<td>Uxxx</td>
<td>standard gate</td>
</tr>
<tr>
<td>UGFF</td>
<td>Uxxx</td>
<td>gated flip-flop</td>
</tr>
<tr>
<td>UIO</td>
<td>Uxxx</td>
<td>digital I/O model</td>
</tr>
<tr>
<td>UTGATE</td>
<td>Uxxx</td>
<td>tristate gate</td>
</tr>
<tr>
<td>VSWITCH</td>
<td>Sxxx</td>
<td>voltage-controlled switch</td>
</tr>
</tbody>
</table>
[tolerance specification]

Appended to each parameter, using the format:

```
[DEV [track&dist] <value>[%]] [LOT [track&dist] <value>[%]]
```

to specify an individual device (DEV) and the device lot (LOT) parameter value deviations. The tolerance specification is used by the `.MC (Monte Carlo analysis)` analysis only.

The LOT tolerance requires that all devices that refer to the same model use the same adjustments to the model parameter. DEV tolerances are independent, that is each device varies independently. The % shows a relative (percentage) tolerance. If it is omitted, `<value>` is in the same units as the parameter itself.

[track & dist]

Specifies the tracking and non-default distribution, using the format:

```
[/<lot #>]/<distribution name>]
```

These specifications must immediately follow the keywords DEV and LOT (without spaces) and are separated by `/`.

<lot #>

Specifies which of ten random number generators, numbered 0 through 9, are used to calculate parameter value deviations. This allows deviations to be correlated between parameters in the same model, as well as between models. The generators for DEV and LOT tolerances are distinct: there are ten generators for DEV tracking and ten generators for LOT tracking. Tolerances without `<lot #>` are assigned individually generated random numbers.

<distribution name>

The distribution name is one of the following. The default distribution can be set by using the DISTRIBUTION parameter of the `.OPTIONS (analysis options)` command.

<table>
<thead>
<tr>
<th>Distribution name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIFORM</td>
<td>Generates uniformly distributed deviations over the range ±&lt;value&gt;.</td>
</tr>
<tr>
<td>GAUSS</td>
<td>Generates deviations using a Gaussian distribution over the range ±3σ and &lt;value&gt; specifies the ±1σ deviation (i.e., this generates deviations greater than ±&lt;value&gt;).</td>
</tr>
<tr>
<td>&lt;user name&gt;</td>
<td>Generates deviations using a user-defined distribution and &lt;value&gt; specifies the ±1 deviation in the user definition; see the <code>.DISTRIBUTION (user-defined distribution)</code> command.</td>
</tr>
</tbody>
</table>

Comments

The examples are for the .MODEL parameter. The last example uses the AKO syntax to reference the parameters of the model QDRIV in the third example.

For more information, refer to your PSpice user’s guide.
Parameters for setting temperature

Some passive and semiconductor devices (C, L, R, B, D, J, M, and Q) have two levels of temperature attributes that can be customized on a model-by-model basis.

First, the temperature at which the model parameters were measured can be defined by using one of the following model parameter formats in the .MODEL command line:

\[
\text{T\_MEASURED} = \langle\text{literal value}\rangle \\
\text{T\_MEASURED} = \{ \langle\text{parameter}\rangle \}
\]

This overrides the nominal TNOM value which is set in the .OPTIONS (analysis options) command line (default = 27°C). All other parameters listed in the .MODEL command are assumed to have been measured at T\_MEASURED.

In addition to the measured model parameter temperature, current device temperatures can be customized to override the circuit’s global temperature specification defined by the .TEMP (temperature) command line (or equivalent .STEP TEMP or .DC TEMP). There are three forms, as described below.

Model parameters for device temperature

<table>
<thead>
<tr>
<th>Description</th>
<th>.MODEL format</th>
<th>Parameter format</th>
<th>Referencing device temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>absolute temperature</td>
<td>standard</td>
<td>T_ABS=\langle value\rangle</td>
<td>T_ABS</td>
</tr>
<tr>
<td>relative to current temperature</td>
<td>standard</td>
<td>T_REL_GLOBAL=\langle value\rangle</td>
<td>global temperature + T_REL_GLOBAL</td>
</tr>
<tr>
<td>relative to AKO model temperature</td>
<td>AKO</td>
<td>T_REL_LOCAL=\langle value\rangle</td>
<td>T_ABS(AKO Model) + T_REL_LOCAL</td>
</tr>
</tbody>
</table>

For all formats, \langle value\rangle can be a literal value or a parameter of the form \{\langle parameter name\rangle\}. A maximum of one device temperature customization can coexist using the T\_MEASURED customization. For example,

\[
\text{.MODEL PNP\_NEW PNP( T\_ABS=35 T\_MEASURED=0 BF=90 )}
\]

defines a new model PNP\_NEW, where BF was measured at 0°C. Any bipolar transistor referencing this model has an absolute device temperature of 35°C.

Examples

One  This example demonstrates device temperatures set relative to the global temperature of the circuit:

\[
\text{.TEMP 10 30 40} \\
\text{.MODEL PNP\_NEW PNP( T\_REL\_GLOBAL=-5 BF=90 )}
\]

This produces three PSpice runs where global temperature changes from 10° to 30° to 40°C, respectively, and any bipolar transistor that references the PNP\_NEW model has a device temperature of 5°, 25°, or 35°C, respectively.
Two

This example sets the device temperature relative to a referenced AKO model:

```
.MODEL PNP_NEW PNP( AKO:PNP_OLD T_REL_LOCAL=10)
.MODEL PNP_OLD PNP( T_ABS=20)
```

Any bipolar transistor referencing the PNP_NEW model has a device temperature of 30°C.

Special considerations

There are a few special considerations when using these temperature parameters:

- If the technique for current device temperature is using the value relative to an AKO model’s absolute temperature (T_ABS), and the AKO referenced model does not specify T_ABS, then the T_REL_LOCAL specification is ignored and the standard global temperature specification is used.

- These temperature parameters cannot be used with the DEV and LOT model parameter tolerance feature.

- A DC sweep analysis can be performed on these parameters so long as the temperature parameter assignment is to a variable parameter. For example:

```
.PARAM PTEMP 27
.MODEL PNP_NEW PNP ( T_ABS={PTEMP} )
.DC PARAM PTEMP 27 35 I
```

This method produces a single DC sweep in PSpice where any bipolar transistor referencing the PNP_NEW model has a device temperature which is swept from 27°C to 35°C in 1°C increments.

A similar effect can be obtained by performing a parametric analysis. For instance:

```
.PARAM PTEMP 27
.MODEL PNP_NEW PNP( T_ABS={PTEMP} )
.STEP PARAM PTEMP 27 35 I
```

This method produces nine PSpice runs where the PNP_NEW model temperature steps from 27°C to 35°C in increments of 1°C, one step per run.

- The effect of a temperature parameter is evaluated once prior to the bias point calculation, unless parameters are swept by means of a .DC PARAM or .STEP PARAM analysis described above. In these cases, the temperature parameter’s effect is reevaluated once for each value of the swept variable.
The .NODESET command helps calculate the bias point by providing an initial best guess for some node voltages and/or inductor currents. Some or all of the circuit’s node voltages and inductor currents can be given the initial guess, and in addition, the voltage between two nodes can be specified.

### General form

```
.NODESET < V(<node> [,<node>])=<value> >*
.NODESET <I(<inductor>)=<value>>
```

### Examples

```
.NODESET V(2)=3.4 V(102)=0 V(3)=-1 V(I(L1)=2uAmp
.NODESET V(InPlus,InMinus)=1e-3 V(100,133)=5.0V
```

### Comments

This command is effective for the bias point (both small-signal and transient bias points) and for the first step of the DC sweep. It has no effect during the rest of the DC sweep, nor during a transient analysis.

Unlike the .IC (initial bias point condition) command, .NODESET provides only an initial guess for some initial values. It does not clamp those nodes to the specified voltages. However, by providing an initial guess, .NODESET can be used to break the tie in, for instance, a flip-flop, and make it come up in a required state.

If both the .IC command and .NODESET command are present, the .NODESET command is ignored for the bias point calculations (.IC overrides .NODESET).

For Capture-based designs, refer to your PSpice user’s guide for more information on setting initial conditions.
.NOISE (noise analysis)

Purpose
The .NOISE command performs a noise analysis of the circuit.

General form
.NOISE V(<node> [,<node>]) <name> [ interval value ]

Examples
.NOISE V(5) VIN
.NOISE V(101) VSRC 20
.NOISE V(4,5) ISRC
.NOISE V([OUT1],[OUT2]) V1

Arguments and options

V(<node> [,<node>])
Output voltage. It has a form such as V(5), which is the voltage at the output node five, or a form such as V(4,5), which is the output voltage between two nodes four and five.

<name>
The name of an independent voltage or current source where the equivalent input noise is calculated. The <name> is not itself a noise generator, but only a place where the equivalent input noise is calculated.

[ interval value ]
Integer that specifies how often the detailed noise analysis data is written to the output file.

Comments
A noise analysis is performed in conjunction with an AC sweep analysis and requires an .AC (AC analysis) command. When .NOISE is used, noise data is recorded in the Probe .DAT file for each frequency in the AC sweep.

The simulator computes:
• Device noise for every resistor and semiconductor in the circuit (propagated to a specified output node)
• Total input and output noise

At each frequency, each noise generator’s contribution is calculated and propagated to the output node. At that point, all the propagated noise values are RMS-summed to calculate the total output noise. The gain from the input source to the output voltage, the total output noise, and the equivalent input noise are all calculated.

For more information, refer to the AC Analyses chapter of your PSpice user’s guide.

If:

<name> is a voltage source
then:
the input noise units are volt/hertz

If:

<name> is a current source
then:
the input noise units are amp/hertz

The output noise units are always volt/hertz.
Every nth frequency, where n is the print interval, a detailed table is printed showing the individual contributions of all the circuit’s noise generators to the total noise. These values are the noise amounts propagated to the output node, not the noise amounts at each generator. If [interval value] is not present, then no detailed table is printed.

The detailed table is printed while the analysis is being performed and does not need a .PRINT (print) command or a .PLOT (plot) command. The output noise and equivalent input noise can be printed in the output by using a .PRINT command or a .PLOT command.
**.OP (bias point)**

**Purpose**  
The .OP command causes detailed information about the bias point to be printed.

**General form**  
.OP

**Examples**  
.OP

**Comments**  
This command does not write output to the Probe data file. The bias point is calculated regardless of whether there is a .OP command. Without the .OP command, the only information about the bias point in the output is a list of the node voltages, voltage source currents, and total power dissipation.

Using a .OP command can cause the small-signal (linearized) parameters of all the nonlinear controlled sources and all the semiconductor devices to be printed in the output file.

The .OP command controls the output for the regular bias point only. The **.TRAN (transient analysis)** command controls the output for the transient analysis bias point.

If no other analysis is performed, then no Probe data file is created.
.OPTIONS (analysis options)

Purpose
The .OPTIONS command is used to set all the options, limits, and control parameters for the simulator.

General form
.OPTIONS [option name]* [ <option name>=<value> ]*

Examples
.OPTIONS NOECHO NOMOD DEFL=12u DEFW=8u DEFAD=150p
+ DEFAS=150p
.OPTIONS ACCT RELTOL=.01
.OPTIONS DISTRIBUTION=GAUSS
.OPTIONS DISTRIBUTION=USERDEF1

Comments
The options can be listed in any order. There are two kinds of options: those with values, and those without values. Options without values are flags that are activated by simply listing the option name.

The .OPTIONS command is cumulative. That is, if there are two (or more) of the .OPTIONS command, the effect is the same as if all the options were listed together in one .OPTIONS command. If the same option is listed more than once, only its last value is used.

For SPICE options not available in PSpice, see Differences between PSpice and Berkeley SPICE2.

Flag options
The default for any flag option is off or no (i.e., the opposite of specifying the option). Flag options affect the output file unless otherwise specified.

<table>
<thead>
<tr>
<th>Flag option</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCT</td>
<td>Summary and accounting information is printed at the end of all the analyses (refer to your PSpice user’s guide for further information on ACCT).</td>
</tr>
<tr>
<td>EXPAND</td>
<td>Lists devices created by subcircuit expansion and lists contents of the bias point file; see .SAVEBIAS (save bias point to file) and .LOADBIAS (load bias point file).</td>
</tr>
<tr>
<td>LIBRARY</td>
<td>Lists lines used from library files.</td>
</tr>
<tr>
<td>LIST</td>
<td>Lists a summary of the circuit elements (devices).</td>
</tr>
<tr>
<td>NOBIAS</td>
<td>Suppresses the printing of the bias point node voltages.</td>
</tr>
<tr>
<td>NODE</td>
<td>Lists a summary of the connections (node table).</td>
</tr>
<tr>
<td>NOECHO</td>
<td>Suppresses a listing of the input file(s).</td>
</tr>
<tr>
<td>Flag option (continued)</td>
<td>Meaning (continued)</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>NOICTRANSLATE</td>
<td>Suppresses the translation of initial conditions (IC attributes) specified on capacitors and inductors into .IC statements (IC pseudocomponents). This means that IC attributes are ignored if the keyword Skip Bias Point (SKIPBP) is not put at the end of the .TRAN statement. See .TRAN (transient analysis).</td>
</tr>
<tr>
<td>NOMOD</td>
<td>Suppresses listing of model parameters and temperature updated values.</td>
</tr>
<tr>
<td>NOOUTMSG</td>
<td>Suppresses simulation error messages in output file.</td>
</tr>
<tr>
<td>NOPAGE</td>
<td>Suppresses paging and the banner for each major section of output.</td>
</tr>
<tr>
<td>NOPRBMSG</td>
<td>Suppresses simulation error messages in Probe data file.</td>
</tr>
<tr>
<td>NOREUSE</td>
<td>Suppresses the automatic saving and restoring of bias point information between different temperatures, Monte Carlo runs, worst-case runs, and .STEP (parametric analysis). See also .SAVEBIAS (save bias point to file) and .LOADBIAS (load bias point file).</td>
</tr>
<tr>
<td>OPTS</td>
<td>Lists values for all options.</td>
</tr>
<tr>
<td>STEPGMIN</td>
<td>Enables GMIN stepping. This causes a GMIN stepping algorithm to be applied to circuits that fail to converge. GMIN stepping is applied first, and if that fails, the simulator falls back to supply stepping.</td>
</tr>
</tbody>
</table>

Option with a name as its value

The following option has a name as its value.

<table>
<thead>
<tr>
<th>Option</th>
<th>Meaning</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISTRIBUTION</td>
<td>default distribution for Monte Carlo deviations</td>
<td>UNIFORM</td>
</tr>
</tbody>
</table>

Default distribution values

The default distribution is used for all of the deviations throughout the Monte Carlo analyses, unless specifically overridden for a particular tolerance. The default value for the default distribution is UNIFORM, but can also be set to GAUSS or to a user-defined (<user name>) distribution. If a user-defined distribution is selected (as illustrated in the last example on page 1-59), a DISTRIBUTION (user-defined distribution) command must be included in the circuit file to define the user distribution for the tolerances. An example would be:

.DISTRIBUTION USERDEF1 (-1.1) (.5,1) (.5,0) (1,0)
### Numerical options with their default values

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTOL</td>
<td>best accuracy of currents</td>
<td>amp</td>
<td>1.0 pA</td>
</tr>
<tr>
<td>CHGTOL</td>
<td>best accuracy of charges</td>
<td>coulomb</td>
<td>0.01 pC</td>
</tr>
<tr>
<td>CPTIME†</td>
<td>CPU time allowed for this run</td>
<td>sec</td>
<td>0.0**</td>
</tr>
<tr>
<td>DEFAD</td>
<td>MOSFET default drain area (AD).</td>
<td>meter²</td>
<td>0.0</td>
</tr>
<tr>
<td>DEFAS</td>
<td>MOSFET default source area (AS).</td>
<td>meter²</td>
<td>0.0</td>
</tr>
<tr>
<td>DEFL</td>
<td>MOSFET default length (L).</td>
<td>meter</td>
<td>100.0 u</td>
</tr>
<tr>
<td>DEFW</td>
<td>MOSFET default width (W).</td>
<td>meter</td>
<td>100.0 u</td>
</tr>
<tr>
<td>DIGFREQ</td>
<td>minimum digital time step is 1/DIGFREQ</td>
<td>hertz</td>
<td>10.0 GHz</td>
</tr>
<tr>
<td>DIGDRVF</td>
<td>minimum drive resistance (Input/Output UIO type model, DRVH (high) and DRVL (low) values)</td>
<td>ohm</td>
<td>2.0</td>
</tr>
<tr>
<td>DIGDRVZ</td>
<td>maximum drive resistance (UIO type model, DRVH and DRVL values)</td>
<td>ohm</td>
<td>20K</td>
</tr>
<tr>
<td>DIGERRDEFAULT</td>
<td>default error limit per digital constraint device</td>
<td></td>
<td>20.0</td>
</tr>
<tr>
<td>DIGERRLIMIT</td>
<td>maximum digital error message limit</td>
<td></td>
<td>0**</td>
</tr>
<tr>
<td>DIGINITSTATE</td>
<td>sets initial state of all flip-flops and latches in circuit: 0=clear, 1=set, 2=X</td>
<td></td>
<td>2.0</td>
</tr>
<tr>
<td>DIGIOLVL</td>
<td>default digital I/O level: 1-4; see UIO in .MODEL (model definition)</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>DIGMNTYMX***</td>
<td>default delay selector: 1=min, 2-typical, 3=max, 4=min/max</td>
<td></td>
<td>2.0</td>
</tr>
<tr>
<td>DIGMNTYSscale</td>
<td>scale factor used to derive minimum delays from typical delays</td>
<td></td>
<td>0.4</td>
</tr>
<tr>
<td>DIGOVDRV</td>
<td>ratio of drive resistances required to allow one output to override another driving the same node</td>
<td></td>
<td>3.0</td>
</tr>
<tr>
<td>DIGTMXSCALE</td>
<td>scale factor used to derive maximum delays from typical delays</td>
<td></td>
<td>1.6</td>
</tr>
<tr>
<td>GMIN</td>
<td>minimum conductance used for any branch</td>
<td>ohm⁻¹</td>
<td>1.0E-12</td>
</tr>
<tr>
<td>ITL1</td>
<td>DC and bias point blind repeating limit</td>
<td></td>
<td>150.0</td>
</tr>
<tr>
<td>ITL2</td>
<td>DC and bias point educated guess repeating limit</td>
<td></td>
<td>20.0</td>
</tr>
<tr>
<td>ITL4</td>
<td>the limit at any repeating point in transient analysis</td>
<td></td>
<td>10.0</td>
</tr>
<tr>
<td>ITL5*</td>
<td>total repeating limit for all points for transient analysis</td>
<td></td>
<td>0.0**</td>
</tr>
</tbody>
</table>

(ITAL5=0 means ITL5=Infinity)
## Numerical options with their default values (continued)

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIMPTS*</td>
<td>maximum points allowed for any print table or plot (LIMPTS=0 means LIMPTS=∞)</td>
<td></td>
<td>0.0**</td>
</tr>
<tr>
<td>NUMDGT</td>
<td>number of digits output in print tables (maximum of 8 useful digits)</td>
<td></td>
<td>4.0</td>
</tr>
<tr>
<td>PIVREL*</td>
<td>relative magnitude required for pivot in matrix solution</td>
<td></td>
<td>1.0E-3</td>
</tr>
<tr>
<td>PIVTOL*</td>
<td>absolute magnitude required for pivot in matrix solution</td>
<td></td>
<td>1.0E-13</td>
</tr>
<tr>
<td>RELTOL</td>
<td>relative accuracy of V and I</td>
<td></td>
<td>0.001</td>
</tr>
<tr>
<td>TNOM</td>
<td>default nominal temperature (also the temperature at which model parameters are assumed to have been measured)</td>
<td>°C</td>
<td>27.0</td>
</tr>
<tr>
<td>VNTOL</td>
<td>best accuracy of voltages</td>
<td>volt</td>
<td>1.0 uV</td>
</tr>
<tr>
<td>WIDTH</td>
<td>same as the .WIDTH OUT= statement (can be set to either 80 or 132)</td>
<td></td>
<td>80.0</td>
</tr>
</tbody>
</table>

*These options are available for modification in PSpice, but it is recommended that the program defaults be used.

**For these options zero means infinity.

***Setting the DIGMNTYMX=4 (min/max) directs PSpice to perform digital worst-case timing simulation. Refer to your PSpice user’s guide for a complete description.
PSpice A/D digital simulation condition messages

Other PSpice features produce warning messages in simulations (e.g., for the digital CONSTRAINT devices monitoring timing relationships of digital nodes). These messages are directed to the PSpice output file (and in Windows, to the Probe data file).

You can use options to control where and how many of these messages are generated. Below is a summary of the PSpice message types and a brief description of their meaning. The condition messages are specific to digital device timing violations and digital worst-case timing hazards. Refer to the Digital Simulation chapter of your PSpice user’s guide for more information on digital worst-case timing.

<table>
<thead>
<tr>
<th>Message type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timing violations</strong></td>
<td></td>
</tr>
<tr>
<td>FREQUENCY</td>
<td>The minimum or maximum frequency specification for a signal has not been satisfied. Minimum frequency violations show that the period of the measured signal is too long, while maximum frequency violations describe signals changing too rapidly.</td>
</tr>
<tr>
<td>GENERAL</td>
<td>A boolean expression described within the GENERAL constraint checker was evaluated and produced a true result.</td>
</tr>
<tr>
<td>HOLD</td>
<td>The minimum time required for a data signal to be stable after the assertion of a clock, has not been met.</td>
</tr>
<tr>
<td>SETUP</td>
<td>The minimum time required for a data signal to be stable prior to the assertion of a clock, has not been met.</td>
</tr>
<tr>
<td>RELEASE</td>
<td>The minimum time for a signal that has gone inactive (usually a control such as CLEAR) to remain inactive before the asserting clock edge, has not been met.</td>
</tr>
<tr>
<td>WIDTH</td>
<td>The minimum pulse width specification for a signal has not been satisfied. That is, a pulse that is too narrow was observed on the node.</td>
</tr>
<tr>
<td><strong>Hazards</strong></td>
<td></td>
</tr>
<tr>
<td>AMBIGUITY CONVERGENCE</td>
<td>The convergence of conflicting rising and falling states (timing ambiguities) arriving at the inputs of a primitive, have produced a pulse (glitch) on the output.</td>
</tr>
<tr>
<td>CUMULATIVE AMBIGUITY</td>
<td>Signal ambiguities are additive, increased by propagation through each level of logic in the circuit. When the ambiguities associated with both edges of a pulse increase to the point where they would overlap, this is flagged as a cumulative ambiguity hazard.</td>
</tr>
<tr>
<td>DIGITAL INPUT VOLTAGE</td>
<td>When a voltage is out of range on a digital pin, PSpice uses the state whose voltage range is closest to the input voltage and continues using the simulation. A warning message is reported.</td>
</tr>
<tr>
<td>NET-STATE CONFLICT</td>
<td>When two or more outputs attempt to drive a net to different states, PSpice represents the conflict as an X (unknown) state. This usually results from improper selection of a bus driver’s enable inputs.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SUPPRESSED GLITCH</td>
<td>A pulse applied to the input of a primitive that is shorter than the active propagation delay is ignored by PSpice. This can or cannot be significant, depending upon the nature of the circuit. The reporting of the suppressed glitch hazard shows that there might be a problem with either the stimulus, or the path delay configuration of the circuit.</td>
</tr>
<tr>
<td>PERSISTENT HAZARD</td>
<td>If the effects of any of the other logic hazard messages mentioned in the output file are able to propagate to either an EXTERNAL port, or to any storage device in the circuit, they are flagged as PERSISTENT HAZARDS. (Refer to your PSpice user’s guide for more details on PERSISTENT HAZARDS.)</td>
</tr>
<tr>
<td>ZERO-DELAY-Oscillation</td>
<td>If the output of a primitive changes more than 50 times within a single digital time step, the node is considered to be oscillating. PSpice reports this and cancels the run.</td>
</tr>
</tbody>
</table>
.PARAM (parameter)

Purpose
The .PARAM statement defines the value of a parameter. A parameter name can be used in place of most numeric values in the circuit description. Parameters can be constants, or expressions involving constants, or a combination of these, and they can include other parameters.

General form
.PARAM <name> = <value> *
.PARAM <name> = { <expression> } *

Examples
.PARAM VSUPPLY = 5V
.PARAM VCC = 12V, VEE = -12V
.PARAM BANDWIDTH = {100kHz/3}
.PARAM PI = 3.14159, TWO.PI = {2*3.14159}
.PARAM VNUM = {2*TWO.PI}

Arguments and options

<name>
Cannot begin with a number, and it cannot be one of the following predefined parameters, or TIME, or .TEXT (text parameter) names.

There are several predefined parameters. The parameter values must be either constants or expressions:

<table>
<thead>
<tr>
<th>Predefined parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEMP</td>
<td>temperature (works using ABM expressions and digital models only)</td>
</tr>
<tr>
<td>VT</td>
<td>thermal voltage (reserved)</td>
</tr>
<tr>
<td>GMIN</td>
<td>shunt conductance for semiconductor p-n junctions</td>
</tr>
</tbody>
</table>

<value>
Constants (<value>) do not need braces { }.

<expression>
Can contain constants or parameters.

Comments
The .PARAM statements are order independent. They can be used inside a subcircuit definition to create local subcircuit parameters. Once defined, a parameter can be used in place of almost all numeric values in the circuit description with the following exceptions:

- Not in the in-line temperature coefficients for resistors (parameters can be used for the TC1 and TC2 resistor model parameters).
- Not in the PWL values for independent voltage and current source (V and I device) parameters.
- Not the E, F, G, and H device SPICE2G6 syntax for polynomial coefficient values and gain.

A .PARAM command can be in a library. The simulator can search libraries for parameters not defined in the circuit file, in the same way it searches for undefined models and subcircuits.

Parameters cannot be used in place of node numbers, nor can the values on an analysis command (e.g., TRAN and AC) be parameterized.
**.PLOT (plot)**

**Purpose**
The .PLOT command causes results from DC, AC, noise, and transient analyses to be line printer plots in the output file.

- This command is included for backward compatibility with earlier versions of PSpice. It is more effective to print plots from within Probe. Printing from Probe yields higher-resolution graphics and provides an opportunity to preview the plot before printing.

**General form**
```
.PLOT <analysis type> [output variable]* + ( [<lower limit value> , <upper limit value>] )*
```

**Examples**
- .PLOT DC V(3) V(2,3) V(R1) I(VIN) I(R2) IB(Q13) VBE(Q13)
- .PLOT AC VM(2) VP(2) VM(3,4) VG(5) VDB(5) IR(D4)
- .PLOT NOISE INOISE ONOISE DB(INOISE) DB(ONOISE)
- .PLOT TRAN V(3) V(2,3) (0,5V) ID(M2) I(VCC) (-50mA,50mA)
- .PLOT TRAN D(QA) D(QB) V(3) V(2,3)
- .PLOT TRAN V(3) V(R1) V([RESET])

**Arguments and options**

- **<analysis type>**
  - DC, AC, NOISE, or TRAN. Only one analysis type can be specified.

- **<output variable>**
  - Following the analysis type is a list of the output variables and (possibly) Y axis scales. A maximum of 8 output variables are allowed on one .PLOT command. However, an analysis can have any number of a .PLOT command. See .PROBE (Probe) for the syntax of the output variables.

- **(<lower limit value>, <upper limit value>)**
  - Sets the range of the y-axis. This forces all output variables on the same y-axis to use the specified range.

  - The same form, (<lower limit value>, <upper limit value>), can also be inserted one or more times in the middle of a set of output variables. Each occurrence defines one Y axis that has the specified range. All the output variables that come between it and the next range to the left in the .PLOT command are put on its corresponding Y axis.

**Comments**
Plots are made by using text characters to draw the plot, which print on any kind of printer. However, plots printed from within Probe look much better.

- The range and increment of the x-axis is fixed by the analysis being plotted. The y-axis default range is determined by the ranges of the output variables. In the fourth example, the two voltage outputs go on the y-axis using the range (0,5V) and the two current outputs go on the y-axis using the range (-5mMA, 50mA).

- Lower and upper limit values do not apply to AC Analysis.
If the different output variables differ considerably in their output ranges, then the plot is given more than one y-axis using ranges corresponding to the different output variables.

The y-axis of frequency response plots (AC) is always logarithmic.

The last example illustrates how to plot the voltage at a node that has a name rather than a number. The first item to plot is a node voltage, the second item is the voltage across a resistor, and the third item is another node voltage, even though the second and third items both begin with the letter R. The square brackets force the interpretation of names to mean node names.
.PRINT (print)

Purpose

The .PRINT command allows results from DC, AC, noise, and transient analyses to be an output in the form of tables, referred to as print tables in the output file.

General form

```
.PRINT[/DGTLCHG] <analysis type> [output variable]*
```

Examples

```
.PRINT DC V(3) V(2,3) V(R1) I(VIN) I(R2) IB(Q13) VBE(Q13)
.PRINT AC VM(2) VP(2) VM(3,4) VG(5) VDB(5) IR(6) II(7)
.PRINT NOISE INOISE ONOISE DB(INOISE) DB(ONOISE)
.PRINT TRAN V(3) V(2,3) ID(M2) I(VCC)
.PRINT TRAN D(QA) D(QB) V(3) V(2,3)
.PRINT/DGTLCHG TRAN QA QB RESET
.PRINT TRAN V(3) V(R1) V([RESET])
```

The last example illustrates how to print a node that has a name, rather than a number. The first item to print is a node voltage, the second item is the voltage across a resistor, and the third item to print is another node voltage, even though the second and third items both begin with the letter R. The square brackets force the names to be interpreted as node names.

Arguments and options

```
[/DGTLCHG]
```

For digital output variables only. Values are printed for each output variable whenever one of the variables changes.

```
<analysis type>
```

Only one analysis type—DC, AC, NOISE, or TRAN—can be specified for each .PRINT command.

```
<output variable>
```

Following the analysis type is a list of the output variables. There is no limit to the number of output variables: the printout is split up depending on the width of the data columns (set using NUMDGT option) and the output width (set using WIDTH option). See .PROBE (Probe) for the syntax of output variables.

Comments

The values of the output variables are printed as a table where each column corresponds to one output variable. You can change the number of digits printed for analog values by using the NUMDGT option of the .OPTIONS (analysis options) command.

An analysis can have multiple .PRINT commands.
.PROBE (Probe)

Purpose
The .PROBE command writes the results from DC, AC, and transient analyses to a data file used by Probe.

General form
.PROBE[/CSDF][output variable]*

Examples
.PROBE
.PROBE V(3) V(2,3) V(R1) I(VIN) I(R2) I(B(Q13)) VBE(Q13)
.PROBE/CSDF
.PROBE V(3) V(R1) V([RESET])
.PROBE D(QBAR)

The first example (with no output variables) writes all the node voltages and all the device currents to the data file. The list of device currents written is the same as the device currents allowed as output variables.

The second example writes only those output variables specified to the data file, to restrict the size of the data file.

The third example creates a data file in a text format using the Common Simulation Data File (CSDF) format, not a binary format. This format is used for transfers between different computer families. CSDF files are larger than regular text files.

The fourth example illustrates how to specify a node that has a name rather than a number. The first item to output is a node voltage, the second item is the voltage across a resistor, and the third item to output is another node voltage, even though the second and third items both begin with the letter R. The square brackets force the interpretation of names to mean node names.

The last example writes only the output at digital node QBAR to the data file, to restrict the size of the data file.

Arguments and options
[output variable]

This section describes the types of output variables allowed in a .PRINT (print), .PLOT (plot), and .PROBE command. Each .PRINT or .PLOT can have up to 8 output variables. This format is similar to that used when calling up waveforms while running Probe.

See the tables below for descriptions of the possible output variables. If .PROBE is used without specifying a list of output variables, all of the circuit voltages and currents are stored for post-processing. When an output variable list is included, the data stored is limited to the listed items. This form is intended for users who want to limit the size of the Probe data file.

Comments
Refer to your PSpice user’s guide for a description of Probe, for information about using the Probe data file, and for more information on the use of text files in Probe. You can also consult Probe Help.

Unlike the .PRINT and .PLOT commands, there are no analysis names before the output variables. Also, the number of output variables is unlimited.
DC Sweep and transient analysis output variables

For DC sweep and transient analysis, these are the available output variables:

<table>
<thead>
<tr>
<th>General form</th>
<th>Meaning of output variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>D(&lt;name&gt;)</td>
<td>digital value of &lt;name&gt; (a digital node) *</td>
</tr>
<tr>
<td>I(&lt;name&gt;)</td>
<td>current through a two terminal device</td>
</tr>
<tr>
<td>Ix(&lt;name&gt;)</td>
<td>current into a terminal of a three or four terminal device</td>
</tr>
<tr>
<td>Iz(&lt;name&gt;)</td>
<td>current into one end of a transmission line (z is either A or B)</td>
</tr>
<tr>
<td>V(&lt;node&gt;)</td>
<td>voltage at a node</td>
</tr>
<tr>
<td>V(&lt;+ node&gt;, &lt;- node&gt;)</td>
<td>voltage between two nodes</td>
</tr>
<tr>
<td>V(&lt;name&gt;)</td>
<td>voltage across a two-terminal device</td>
</tr>
<tr>
<td>Vx(&lt;name&gt;)</td>
<td>voltage at a non-grounded terminal of a device (see Ix)</td>
</tr>
<tr>
<td>Vz(&lt;name&gt;)</td>
<td>voltage at one end of a transmission line (z is either A or B)</td>
</tr>
<tr>
<td>Vxy(&lt;name&gt;)</td>
<td>voltage across two terminals of a three or four terminal device type</td>
</tr>
</tbody>
</table>

*These values are available for transient and DC analysis only. For the .PRINT/DGTLCHG statement, the D() is optional.

<table>
<thead>
<tr>
<th>Example</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>D(QA)</td>
<td>the value of digital node QA</td>
</tr>
<tr>
<td>I(D5)</td>
<td>current through diode D5</td>
</tr>
<tr>
<td>I(G10)</td>
<td>current into gate of G10</td>
</tr>
<tr>
<td>V(3)</td>
<td>voltage between node three and ground</td>
</tr>
<tr>
<td>V(3,2)</td>
<td>voltage between nodes three and two</td>
</tr>
<tr>
<td>V(R1)</td>
<td>voltage across resistor R1</td>
</tr>
<tr>
<td>V(A(T2))</td>
<td>voltage at port A of T2</td>
</tr>
<tr>
<td>V(B(Q3))</td>
<td>voltage between base of transistor Q3 and ground</td>
</tr>
<tr>
<td>VGS(M13)</td>
<td>gate-source voltage of M13</td>
</tr>
</tbody>
</table>
Multiple-terminal devices

For the V(<name>) and I(<name>) forms, where <name> must be the name of a two-terminal device, the devices are:

<table>
<thead>
<tr>
<th>Character ID</th>
<th>Two-terminal device</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>capacitor</td>
</tr>
<tr>
<td>D</td>
<td>diode</td>
</tr>
<tr>
<td>E</td>
<td>voltage-controlled voltage source</td>
</tr>
<tr>
<td>F</td>
<td>current-controlled current source</td>
</tr>
<tr>
<td>G</td>
<td>voltage-controlled current source</td>
</tr>
<tr>
<td>H</td>
<td>current-controlled voltage source</td>
</tr>
<tr>
<td>I</td>
<td>independent current source</td>
</tr>
<tr>
<td>L</td>
<td>inductor</td>
</tr>
<tr>
<td>R</td>
<td>resistor</td>
</tr>
<tr>
<td>S</td>
<td>voltage-controlled switch</td>
</tr>
<tr>
<td>V</td>
<td>independent voltage source</td>
</tr>
<tr>
<td>W</td>
<td>current-controlled switch</td>
</tr>
</tbody>
</table>

For the Vx(<name>), Vxy(<name>), and Ix(<name>) forms, where <name> must be the name of a three or four-terminal device and x and y must each be a terminal abbreviation, the devices and the terminals areas follows. For the Vz(<name>) and Iz(<name>) forms, <name> must be the name of a transmission line (T device) and z must be A or B.

<table>
<thead>
<tr>
<th>Three &amp; four-terminal device type</th>
<th>Terminal abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>B (GaAs MESFET)</td>
<td>D (drain)</td>
</tr>
<tr>
<td></td>
<td>G (gate)</td>
</tr>
<tr>
<td></td>
<td>S (source)</td>
</tr>
<tr>
<td>J (Junction FET)</td>
<td>D (drain)</td>
</tr>
<tr>
<td></td>
<td>G (gate)</td>
</tr>
<tr>
<td></td>
<td>S (source)</td>
</tr>
<tr>
<td>M (MOSFET)</td>
<td>D (drain)</td>
</tr>
<tr>
<td></td>
<td>G (gate)</td>
</tr>
<tr>
<td></td>
<td>S (source)</td>
</tr>
<tr>
<td></td>
<td>B (bulk, substrate)</td>
</tr>
<tr>
<td>Q (Bipolar transistor)</td>
<td>C (collector)</td>
</tr>
<tr>
<td></td>
<td>B (base)</td>
</tr>
<tr>
<td></td>
<td>E (emitter)</td>
</tr>
<tr>
<td></td>
<td>S (substrate)</td>
</tr>
<tr>
<td>Three &amp; four-terminal device type</td>
<td>Terminal abbreviation</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>T (transmission line)</td>
<td>Va (near side voltage)</td>
</tr>
<tr>
<td></td>
<td>Ia (near side current)</td>
</tr>
<tr>
<td></td>
<td>Vb (far side voltage)</td>
</tr>
<tr>
<td></td>
<td>Ib (far side current)</td>
</tr>
<tr>
<td>Z (IGBT)</td>
<td>C (collector)</td>
</tr>
<tr>
<td></td>
<td>G (gate)</td>
</tr>
<tr>
<td></td>
<td>E (emitter)</td>
</tr>
</tbody>
</table>
AC analysis

For AC analysis, the output variables listed in the preceding section are augmented by adding a suffix.

For AC analysis, the suffixes are ignored for a .PROBE command, but can be used in a .PRINT (print) command and a .PLOT (plot) command, and when adding a trace in Probe. For example, in a .PROBE command, VDB(R1) is translated to V(R1), which is the raw data.

For these devices, you need to put a zero-valued voltage source in series with the device (or terminal) of interest before you can print or plot the current through this voltage source.

### Suffix

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Meaning of output variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>magnitude</td>
</tr>
<tr>
<td>DB</td>
<td>magnitude in decibels</td>
</tr>
<tr>
<td>G</td>
<td>group delay (−dPHASE/dFREQUENCY)</td>
</tr>
<tr>
<td>I</td>
<td>imaginary part</td>
</tr>
<tr>
<td>M</td>
<td>magnitude</td>
</tr>
<tr>
<td>P</td>
<td>phase in degrees</td>
</tr>
<tr>
<td>R</td>
<td>real part</td>
</tr>
</tbody>
</table>

### Examples

<table>
<thead>
<tr>
<th>Examples</th>
<th>Meaning of output variables for AC analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>II(R13)</td>
<td>imaginary part of current through R13</td>
</tr>
<tr>
<td>IGG(M3)</td>
<td>group delay of gate current for M3</td>
</tr>
<tr>
<td>IR(VIN)</td>
<td>real part of I through VIN</td>
</tr>
<tr>
<td>IAG(T2)</td>
<td>group delay of current at port A of T2</td>
</tr>
<tr>
<td>V(2,3)</td>
<td>magnitude of complex voltage across nodes 2 &amp; 3</td>
</tr>
<tr>
<td>VDB(R1)</td>
<td>db magnitude of V across R1</td>
</tr>
<tr>
<td>VBEP(Q3)</td>
<td>phase of base-emitter V at Q3</td>
</tr>
<tr>
<td>VM(2)</td>
<td>magnitude of V at node 2</td>
</tr>
</tbody>
</table>

Current outputs for the F and G devices are not available for DC and transient analyses.
Noise analysis

For noise analysis, the output variables are predefined as follows:

<table>
<thead>
<tr>
<th>Output variable</th>
<th>Meaning of output variables for noise analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>INoise</td>
<td>Total RMS summed noise at input node</td>
</tr>
<tr>
<td>ONoise</td>
<td>INoise equivalent at output node</td>
</tr>
<tr>
<td>DB(INoise)</td>
<td>INoise in decibels</td>
</tr>
<tr>
<td>DB(ONoise)</td>
<td>ONoise in decibels</td>
</tr>
</tbody>
</table>

`.PRINT (print)` and `.PLOT (plot)` cannot be used for the noise from any one device. However, the print interval on the `.NOISE (noise analysis)` command can be used to output this information.
The .SAVEBIAS command saves the bias point node voltages and inductor currents, to a file. It is used concurrently with .LOADBIAS (load bias point file).

Only one analysis is specified in a .SAVEBIAS command, which can be OP, TRAN, or DC. However, a circuit file can contain a .SAVEBIAS command for each of the three analysis types. If the simulation parameters do not match the keywords and values in the .SAVEBIAS command, then no file is produced.

**General form**
```
.SAVEBIAS <"file_name"> <[OP] [TRAN] [DC]> [NOSUBCKT]
+[TIME=<value> [REPEAT]] [TEMP=<value>]
+ [STEP=<value>] [MCRUN=<value>] [DC=<value>]
+ [DC1=<value>] [DC2=<value>]
```

**Examples**
```
.SAVEBIAS "OPPOINT" OP
For the first example, the small-signal operating point (.AC or .OP) bias point is saved.

.SAVEBIAS "TRANDATA.BSP" TRAN NOSUBCKT TIME=10u
In the second example, the transient bias point is written out at the time closest to, but not less than 10.0 u/sec. No bias point information for subcircuits is saved.

.SAVEBIAS "SAVETRAN.BSP" TRAN TIME=5n REPEAT TEMP=50.0
Use of the [REPEAT] keyword in the third example causes the bias point to be written out every 5.0 ns when the temperature of the run is 50.0 degrees.

.SAVEBIAS "DCBIAS.SAV" DC
In the fourth example, because there are no parameters supplied, only the very first DC bias point is written to the file.

.SAVEBIAS "SAVEDC.BSP" DC MCRUN=3 DC1=3.5 DC2=100
The fifth example saves the DC bias point when the following three conditions are all met: the first DC sweep value is 3.5, the second DC sweep value is 100, and the simulation is on the third Monte Carlo run. If only one DC sweep is being performed, then the keyword DC can be substituted for DC1.

**Arguments and options**

`<"file_name">`
Any valid file name for the computer system, which must be enclosed in quotation marks.

[NOSUBCKT]
When used, the node voltages and inductor currents for subcircuits are not saved.

[TIME=<value> [REPEAT]]
Used to define the transient analysis time at which the bias point is to be saved.

[TEMP=<value>]
Defines the temperature at which the bias point is to be saved. [STEP=<value>]
The step value at which the bias point is to be saved.
Commands

.SAVEBIAS (save bias point to file)

[ MCRUN=<value> ]
   The number of the Monte Carlo or worst-case analysis run for which the bias point is to 
   be saved.

[ DC=<value> ], [ DC1=<value> ], and [ DC2=<value> ]
   Used to specify the DC sweep value at which the bias point is to be saved.

Comments

If REPEAT is not used, then the bias at the next time point greater than or equal to 
TIME=<value> is saved. If REPEAT is used, then TIME=<value> is the interval at which the 
bias is saved. However, only the latest bias is saved; any previous times are overwritten. The 
[ TIME=<value> [ REPEAT ] ] can only be used with a transient analysis.

The [ DC=<value> ] should be used if there is only one sweep variable. If there are two sweep 
variables, then [ DC1=<value> ] should be used to specify the first sweep value and 
[ DC2=<value> ] should be used to specify the second sweep value.

The saved bias point information is in the following format: one or more comment lines that 
list items such as:
   • circuit name, title, date and time of run, analysis, and temperature, or 
   • a single .NODESET (set approximate node voltage for bias point) command 
     containing the bias point voltage values and inductor currents.

Only one bias point is saved to the file during any particular analysis. At the specified time, 
the bias point information and the operating point data for the active devices and controlled 
sources are written to the output file. When the supplied specifications on the .SAVEBIAS 
command line match the state of the simulator during execution, the bias point is written out.

Usage examples

A .SAVEBIAS command and a .LOADBIAS (load bias point file) command can be used to 
shorten the simulation time of large circuits, and also to aid in convergence.

A typical application for a .SAVEBIAS and a .LOADBIAS command is for a simulation that 
takes a considerable amount of time to converge to a bias point. The bias point is saved using 
a .SAVEBIAS command so that when the simulation is run again, the previous bias point 
calculated is used as a starting point for the bias solution, to save processing time.

The following example illustrates this procedure for a transient simulation.

.SAVEBIAS "SAVEFILE TRN" TRAN

When the simulation is run, the transient analysis bias point information is saved to the file 
savefile.trn in the form of a .NODESET command. This .NODESET command provides 
the simulator with a starting solution for determining the bias point calculation for future 
simulations. To use this file, replace the .SAVEBIAS command in the circuit file using the 
following .LOADBIAS (Load Bias Point File) command.

.LOADBIAS "SAVEFILE TRN"

A .SAVEBIAS and .LOADBIAS command should not refer to the same file during the 
same simulation run. Use the .SAVEBIAS during the first simulation and the 
 LOADBIAS for subsequent ones.

The simulator algorithms have been changed to provide an automatic saving and loading of 
bias point information under certain conditions. This automatic feature is used in the
Commands

**.SAVEBIAS** (save bias point to file)

The **.SAVEBIAS** command is used to save the bias point information to a file. This can be useful for transient analysis where the bias point is calculated at several temperatures. For example, the command `TEMP 0 10 20 30` can be used to specify temperatures, and the bias point is calculated at each of these temperatures.

A typical application is a transient analysis where the bias point is calculated at several temperatures (such as `TEMP 0 10 20 30`). As each new temperature is processed, the bias point for the previous temperature is used to find the new bias point. Since this process is automatic, the user does not have to change anything in the circuit file. However, there is some memory overhead since the bias point information is saved during the simulation. Disable the automatic saving feature, using the NOREUSE flag option in the **.OPTIONS** (analysis options) command as follows:

```
.OPTIONS NOREUSE
```

Another application for the **.LOADBIAS** and **.SAVEBIAS** command is the handling of convergence problems. Consider a circuit which has difficulty in starting a DC sweep. The designer has added a **.NODESET** command as shown below to help the simulator determine the bias point solution.

```
.NODESET V(3)=5.0V V(4)=2.75V
```

Even though this helps the simulator determine the bias point, the simulator still has to compute the starting values for each of the other nodes. These values can be saved using the following statement:

```
.SAVEBIAS "DCOP.NOD" DC
```

The next time the simulation is run, the **.NODESET** and **.SAVEBIAS** command should be removed and replaced using the following:

```
.LOADBIAS "DCOP.NOD"
```

This provides the starting values for all of the nodes in the circuit, and can assist the simulator in converging to the correct bias point for the start of the sweep. If convergence problems are caused by a change in the circuit topology, the designer can edit the bias point save file to change the values for specific nodes or to add new nodes.
.SENS (sensitivity analysis)

Purpose
The .SENS command performs a DC sensitivity analysis.

General form
.SENS <output variable>*

Examples
.SENS V(9) V(4,3) V(17) I(VCC)

Arguments and options

<output variable>
Same format and meaning as in the .PRINT command for DC and transient analyses. However, when <output variable> is a current, it is restricted to be the current through a voltage source.

Comments
By linearizing the circuit about the bias point, the sensitivities of each of the output variables to all the device values and model parameters is calculated and output data generated. This can generate large amounts of output data.

Device sensitivities are only provided for the following device types:
- resistors
- independent voltage and current sources
- voltage and current-controlled switches
- diodes
- bipolar transistors

The results of the .SENS command are only available in the output file. They cannot be viewed in Probe.
.STEP (parametric analysis)

Purpose

The .STEP command performs a parametric sweep for all of the analyses of the circuit. The .STEP command is similar to the .TEMP (temperature) command in that all of the typical analyses—such as .DC (DC analysis), .AC (AC analysis), and .TRAN (transient analysis)—are performed for each step.

Once all the runs finish, the specified .PRINT (Print) table or .PLOT (Plot) plot for each value of the sweep is an output, just as for the .TEMP or .MC (Monte Carlo Analysis) command.

Probe displays nested sweeps as a family of curves.

General form

.SEP LIN <sweep variable name>
+ <start value> <end value> <increment value>

.SEP [DEC |OCT] <sweep variable name>
+ <start value> <end value> <points value>

.SEP <sweep variable name> LIST <value>*

The first general form is for doing a linear sweep. The second form is for doing a logarithmic sweep. The third form is for using a list of values for the sweep variable.

Examples

.SEP VCE 0V 10V .5V
.SEP LIN I2 5mA -2mA 0.1mA
.SEP RES RMOD(R) 0.9 1.1 .001
.SEP DEC NPN QFAST(IS) 1E-18 1E-14 5
.SEP TEMP LIST 0 20 27 50 80 100
.SEP PARAM CenterFreq 9.5kHz 10.5kHz 50Hz

The first three examples are for doing a linear sweep. The fourth example is for doing a logarithmic sweep. The fifth example is for using a list of values for the sweep variable.

Arguments and options

Sweep type

The sweep can be linear, logarithmic, or a list of values. For [linear sweep type], the keyword LIN is optional, but either OCT or DEC must be specified for the <logarithmic sweep type>. The sweep types are described below.

<table>
<thead>
<tr>
<th>Sweep types</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIN</td>
<td>Linear sweep. The sweep variable is swept linearly from the starting to the ending value. The &lt;increment value&gt; is the step size</td>
</tr>
<tr>
<td>OCT</td>
<td>Sweep by octaves. The sweep variable is swept logarithmically by octaves. The &lt;points value&gt; is the number of steps per octave.</td>
</tr>
<tr>
<td>DEC</td>
<td>Sweep by decades. The sweep variable is swept logarithmically by decades. The &lt;points value&gt; is the number of steps per decade.</td>
</tr>
<tr>
<td>LIST</td>
<td>Use a list of values. In this case there are no start and end values. Instead, the numbers that follow the keyword LIST are the values that the sweep variable is set to.</td>
</tr>
</tbody>
</table>
The LIST values must be in either ascending or descending order.

<sweep variable name>

The <sweep variable name> can be one of the types described below.

<table>
<thead>
<tr>
<th>Sweep Variable Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>A name of an independent voltage or current source. During the sweep, the source’s voltage or current is set to the sweep value.</td>
</tr>
<tr>
<td>model parameter</td>
<td>A model type and model name followed by a model parameter name in parenthesis. The parameter in the model is set to the sweep value.</td>
</tr>
<tr>
<td>temperature</td>
<td>Use the keyword TEMP for &lt;sweep variable name&gt;. The temperature is set to the sweep value. For each value in the sweep, all the circuit components have their model parameters updated to that temperature.</td>
</tr>
<tr>
<td>global parameter</td>
<td>Use the keyword PARAM, followed by the parameter name, for &lt;sweep variable name&gt;). During the sweep, the global parameter’s value is set to the sweep value and all expressions are reevaluated.</td>
</tr>
</tbody>
</table>

<start value>

Can be greater or less than <end value>: that is, the sweep can go in either direction.

<increment value> and <points value>

Must be greater than zero.

Comments

The .STEP command is similar to the .DC (DC analysis) command and immediately raises the question of what happens if both .STEP and .DC try to set the same value. The same question can come up using .MC (Monte Carlo analysis). The answer is that this is not allowed: no two analyses (.STEP, .TEMP (temperature), .MC, .WCASE (sensitivity/worst-case analysis), and .DC) can try to set the same value. This is flagged as an error during read-in and no analyses are performed.

You can use the .STEP command to look at the response of a circuit as a parameter varies, for example, how the center frequency of a filter shifts as a capacitor varies. By using .STEP, that capacitor can be varied, producing a family of AC waveforms showing the variation. Another use is for propagation delay in transient analysis.
Usage examples

One  The .STEP command only steps the DC component of an AC source. In order to step the AC component of an AC source, a variable parameter has to be created. For example,

```
Vac 1 0 AC {variable}
.param variable=0
.step param variable 0 5 1
.ac dec 100 1000 1e6
```

Two  This is one way of stepping a resistor from 30 to 50 ohms in steps of 5 ohms, using a global parameter:

```
.PARAM RVAL = 1
R1 1 2 {RVAL}
.STEP PARAM RVAL 30,50,5
```

The parameter RVAL is global and PARAM is the keyword used by the .STEP command when using a global parameter.

Three  The following example steps the resistor model parameter R. This is another way of stepping a resistor from 30 to 50 ohms in steps of 5 ohms.

```
R1 1 2 RMOD 1
.MODEL RMOD RES(R=30)
.STEP RES RMOD(R) 30,50,5
```

Do not use R={30}.

Here RMOD is the model name, RES is the sweep variable name (a model type), and R is the parameter within the model to step. To step the value of the resistor, the line value of the resistor is multiplied by the R parameter value to achieve the final resistance value, that is:

```
final resistor value = line resistor value · R
```

Therefore, if the line value of the resistor is set to one ohm, the final resistor value is 1 · R or R. Stepping R from 30 to 50 ohms then steps the resistor value from 1 · 30 ohms to 1 · 50 ohms.

In examples 2 and 3, all of the ordinary analyses (e.g., .DC, .AC, and .TRAN) are run for each step.
.STIMLIB (stimulus library file)

**Purpose**

The .STIMLIB command makes stimulus library files created by StmEd available to PSpice.

**General form**

```
.STIMLIB <file name>[.stl]
```

**Examples**

```
.STIMLIB mylib.stl
.STIMLIB volts.stl
.STIMLIB dgpulse
```

**Arguments and options**

```
<file name>
```

Specification that identifies a file containing .STIMULUS commands.
.STIMULUS (stimulus)

Purpose

The .STIMULUS command encompasses only the Transient specification portion of what is allowed in the V or I device syntax.

General form

.STIMULUS <stimulus name> <type> <type-specific parameters>*

Examples

.STIMULUS InputPulse PULSE (-1mv 1mv 2ns 2ns 50ns 100ns)
.STIMULUS DigitalPulse STIM (1,1)
+ 0S 1
+ 10NS 0
+ 20NS 1
.STIMULUS 50KHZSIN SIN (0 5 50KHZ 0 0 0)

Arguments and options

<stimulus name>

The name by which the stimulus is referred to by the source devices (V or I), or by the digital STIM device.

---

.STIMULUS commands generally appear within stimulus libraries created by StmEd.
.SUBCKT (subcircuit)

.ENDS (end subcircuit)

Purpose

The .SUBCKT command/statement starts the subcircuit definition by specifying its name, the number and order of its terminals, and the names and default parameters that control its behavior. Subcircuits are instantiated using X (Subcircuit instantiation) devices. The .ENDS command marks the end of a subcircuit definition.

General form

```
.SUBCKT <name> [node]*
+ [OPTIONAL: < <interface node> = <default value> >]*
+ [PARAMS: < <name> = <value> >]*
+ [TEXT: < <name> = <text value> >]*
...
.ENDS
```

Examples

```
.SUBCKT OPAMP 1 2 101 102 17
...
.ENDS

.SUBCKT FILTER INPUT, OUTPUT PARAMS: CENTER=100kHz,
+ BANDWIDTH=10kHz
...
.ENDS

.SUBCKT PLD IN1 IN2 IN3 OUT1
+ PARAMS: MNTYMXDLY=0 IO_LEVEL=0
+ TEXT: JEDEC_FILE="PROG.JED"
...
.ENDS

.SUBCKT 74LS00 A B Y
+ OPTIONAL: DPWR=$G_DPWR DGND=$G_DGND
+ PARAMS: MNTYMXDLY=0 IO_LEVEL=0
...
.ENDS
```

Arguments and options

- `<name>`
  The name is used by an X (Subcircuit Instantiation) device to reference the subcircuit.

- `[node]*`
  An optional list of nodes (pins). This is optional because it is possible to specify a subcircuit that has no interface nodes.

  OPTIONAL:
  Allows specification of one or more optional nodes (pins) in the subcircuit definition.
The subcircuit definition ends with a .ENDS command. All of the netlist between .SUBCKT and .ENDS is included in the definition. Whenever the subcircuit is used by an X (Subcircuit Instantiation) device, all of the netlist in the definition replaces the X device.

There must be the same number of nodes in the subcircuit calling statements as in its definition. When the subcircuit is called, the actual nodes (the ones in the calling statement) replace the argument nodes (the ones in the defining statement).

Do not use 0 (zero) in this node list. Zero is reserved for the global ground node.

The optional nodes are stated as pairs consisting of an interface node and its default value. If an optional node is not specified in an X device, its default value is used inside the subcircuit; otherwise, the value specified in the definition is used.

This feature is particularly useful when specifying power supply nodes, because the same nodes are normally used in every device. This makes the subcircuits easier to use because the same two nodes do not have to be specified in each subcircuit statement. This method is used in the libraries provided with the Digital Simulation feature.

Subcircuits can be nested. That is, an X device can appear between .SUBCKT and .ENDS commands. However, subcircuit definitions cannot be nested. That is, a .SUBCKT statement cannot appear in the statements between a .SUBCKT and a .ENDS.

Subcircuit definitions should contain only device instantiations (statements without a leading period) and possibly these statements:

- **.IC (initial bias point condition)**
- **.NODESET (set approximate node voltage for bias point)**
- **.MODEL (model definition)**
- **.PARAM (parameter)**
- **.FUNC (function)**

Models, parameters, and functions defined within a subcircuit definition are available only within the subcircuit definition in which they appear. Also, if a .MODEL, .PARAM, or a .FUNC statement appears in the main circuit, it is available in the main circuit and all subcircuits.

Node, device, and model names are local to the subcircuit in which they are defined. It is acceptable to use a name in a subcircuit which has already been used in the main circuit. When the subcircuit is expanded, all its names are prefixed using the subcircuit instance name: for example, Q13 becomes X3.Q13 and node 5 becomes X3.5 after expansion. After expansion all names are unique. The only exception is the use of global node names (refer to your PSpice user’s guide) that are not expanded.
The keyword PARAMS: passes values into subcircuits as arguments and uses them in expressions inside the subcircuit. The keyword TEXT: passes text values into subcircuits as arguments and uses them as expressions inside the subcircuit. Once defined, a text parameter can be used in the following places:

- To specify a JEDEC file name on a PLD device.
- To specify an Intel Hex file name to program a ROM device or initialize a RAM device.
- To specify a stimulus file name or signal name on a FSTIM device.
- To specify a text parameter to a (lower level) subcircuit.
- As part of a text expression used in one of the above.

The text parameters and expressions are currently only used in Digital Simulation.

Usage examples

One  In the example of the 74LS00 subcircuit, the following subcircuit reference uses the default power supply nodes $G_DPWR and $G_DGND:

X1 IN1 IN2 OUT 74LS00

Two  To specify your own power supply nodes MYPOWER and MYGROUND, use the following subcircuit instantiation:

X2 IN1 IN2 OUT MYPOWER MYGROUND 74LS00

Three  If wanted, one optional node in the subcircuit instantiation can be provided. In the following subcircuit instantiation, the default $G_DGND would be used:

X3 IN1 IN2 OUT MYPOWER 74LS00

Four  However, to specify values beyond the first optional node, all nodes previous to that node must be specified. For example, to specify your own ground node, the default power node before it must be explicitly stated:

X4 IN1 IN2 OUT $G_DPWR MYGROUND 74LS00
.TEMP (temperature)

**Purpose**
The .TEMP command sets the temperature at which all analyses are done.

**General form**
```
.TEMP <temperature value> *
```

**Examples**
```
.TEMP 125
.TEMP 0 27 125
```

The temperatures are in degrees Centigrade. If more than one temperature is given, then all analyses are performed for each temperature.

It is assumed that the model parameters were measured or derived at the nominal temperature, TNOM (27°C by default). See the `.OPTIONS (analysis options)` command for setting TNOM.

.TEMP behaves similarly to the list variant of the `.STEP (parametric analysis)` statement, with the stepped variable being the temperature.
.TEXT (text parameter)

**Purpose**
The .TEXT command precedes a list of names and text values.

**General form**
```
.TEXT < <name> = "<text value>" >*
.TEXT < <name> = | <text expression> | >*
```

**Examples**
```
.TEXT MYFILE = "FILENAME.EXT"
.TEXT FILE = "ROM.DAT", FILE2 = "ROM2.DAT"
.TEXT PROGDAT = |"ROM"+TEXTINT(RUN_NO)+".DAT"|
.TEXT DATA1 = "PLD.JED", PROGDAT = |"\PROG\DAT"+FILENAME|
```

**Arguments and options**
- `<name>`
  - Cannot be a .PARAM name, or any of the reserved parameters names.
- `<text expression>`
  - Text expressions can contain the following:

<table>
<thead>
<tr>
<th>Text expressions</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>enclosed in “ “</td>
<td>text constants</td>
</tr>
<tr>
<td>text parameters</td>
<td>previously defined parameters</td>
</tr>
<tr>
<td>+</td>
<td>the operator that concatenates two text values</td>
</tr>
<tr>
<td>TEXTINT (&lt;value or expression&gt;)</td>
<td>a function which returns a text string which is the integer value closest to the value of the <code>&lt;value or expression&gt;</code>; (&lt;value or expression&gt; is a floating-point value)</td>
</tr>
</tbody>
</table>

The values can be text constants (enclosed in quotation marks “ “) or text expressions (enclosed in |). Text expressions can contain only text constants or previously defined parameters. Once defined, a text parameter has the following uses:

- To specify a JEDEC file name on a PLD device.
- To specify an Intel Hex file name to program a ROM device or initialize a RAM device.
- To specify a stimulus file name or signal name on an FSTIM device.
- To specify a text parameter to a subcircuit.
- As part of a text expression used in one of the above.

Text parameters and expressions are only used in digital simulation.
.TF (transfer)

Purpose
The .TF command/statement causes the small-signal DC gain to be calculated by linearizing the circuit around the bias point.

General form
.TF <output variable> <input source name>

Examples
.TF V(5) VIN
.TF I(VDRIV) ICNTRL

Arguments and options

<output variable>
This has the same format and meaning as in the .PRINT statement.

The gain from <input source name> to <output variable> and the input and output resistances are evaluated and written to the output file. This output does not require a .PRINT (print), .PLOT (plot), or .PROBE (Probe) statement. When <output variable> is a current, it is restricted to be the current through a voltage source.

The results of the .TF command are only available in the output file. They cannot be viewed in Probe.
.TRAN (transient analysis)

Purpose
The .TRAN command causes a transient analysis to be performed on the circuit and specifies the time period for the analysis.

General form
.TRAN[/OP] <print step value> <final time value> +[no-print value [step ceiling value]][SKIPBP]

Examples
.TRAN 1ns 100ns
.TRAN/OP 1ns 100ns 20ns SKIPBP
.TRAN 1ns 100ns 0ns .1ns

Arguments and options

[/OP]
Causes the same detailed printing of the bias point that the .OP (bias point) command does for the regular bias point. Without using this option, only the node voltages are printed for the transient analysis bias point.

<print step value>
Sets the time interval used for printing (.PRINT), plotting (.PLOT), or performing a Fourier integral on (.FOUR) the results of the transient analysis.

Since the results are computed at different times than they are printed, a 2nd-order polynomial interpolation is used to obtain the printed values. This applies only to .PRINT (print), .PLOT (plot), and .FOUR (Fourier analysis) outputs and does not affect Probe.

<final time value>
Sets the end time for the analysis.

[no-print value]
Sets the time interval (from TIME=0) that is not printed, plotted, or given to Probe.

[step ceiling value]
Overrides the default ceiling on the internal time step with a lower value.

[SKIPBP]
Skips calculation of the bias point.

When this option is used, the bias conditions are fully determined by the IC= specifications for capacitors and inductors.
The transient analysis calculates the circuit’s behavior over time, always starting at TIME=0 and finishing at <final time value>, but you can suppress the output of a portion of the analysis. Use a PRINT (print), PLOT (plot), FOUR (Fourier analysis), or PROBE (Probe) to get the results of the transient analysis.

Prior to performing the transient analysis, PSpice computes a bias point for the circuit separate from the regular bias point. This is necessary because at the start of a transient analysis, the independent sources can have different values than their DC values.

The internal time step of the transient analysis adjusts as the analysis proceeds: over intervals when there is little activity, the time step is increased, and during busy intervals it is decreased. The default ceiling on the internal time step is <final time value>/50, but when there are no charge storage elements, inductances, or capacitances in the circuit, the ceiling is <print step value>.

The .TRAN command also sets the variables TSTEP and TSTOP, which are used in defaulting some waveform parameters. TSTEP is equal to <print step value> and TSTOP is equal to <final time value>.

Refer to your PSpice user’s guide for more information on setting initial conditions.
.VECTOR (digital output)

Purpose
The .VECTOR command creates files containing digital simulation results.

General form
```
.VECTOR <number of nodes> <node>*
+ [ POS = <column position> ]
+ [ FILE = <filename> ]
+ [ RADIX = "Binary" | "Hex" | "Octal"
+ [ BIT = <bit index> ] ]
+ [ SIGNAMES = <signal names> ]
```

Examples
```
.VECTOR 1 CLOCK SIGNAMES=SYSCLK
.VECTOR 4 DATA3 DATA2 DATA1 DATA0
.VECTOR 1 ADDR3 POS=2 RADIX=H BIT=4
.VECTOR 1 ADDR2 POS=2 RADIX=H BIT=3
.VECTOR 1 ADDR1 POS=2 RADIX=H BIT=2
.VECTOR 1 ADDRO POS=2 RADIX=H BIT=1
```

Arguments and options

- `<filename>`
  Specifies the name of the file to which the simulation results are saved. If left blank, the simulator creates a file named `<circuit filename>.vec`, where `<circuit filename>.cir` is the name of the netlist file.

- `<number of nodes>`
  This means the number of nodes in the list.

- `<node>`
  This defines the nodes whose states are to be stored.

- `<column position>`
  Specifies the column position in the file. By default, the column position is determined through the order in which the .VECTOR command appears in the circuit file, and by the order of the signals within a .VECTOR command. Valid values for `<column position>` are 1-255.

- `RADIX`
  The radix of the values for the specified nodes is defined if `<number of nodes>` is greater than one. Valid values are BINARY, OCTAL, or HEX (you can abbreviate to the first letter). If `<number of nodes>` is one, and a radix of OCTAL or HEX is specified, a bit position within the octal or hex digit via the BIT parameter can also be specified. A separate .VECTOR command can be used to construct multi-bit values out of single signals, provided the same POS value is specified. The default radix is BINARY if `<number of nodes>` is one. Otherwise, the default radix is HEX. If a radix of OCTAL or HEX is specified, the simulator creates dummy entries in the vector file header to fill out the value if `<number of nodes>` is not an even power of two.

- `<bit index>`
  Defines the bit position within a single hex or octal digit when the VECTOR symbol is attached to a wire. Valid values are one through four if RADIX=HEX, and one through three if RADIX=OCTAL.
Commands

.VECTOR (digital output)

<signal names>
  Defines the names of the signals which appear in the header of the vector file. If
  SIGNAMES is not specified, the <node> names are used in the vector file header. If
  <number of nodes> is greater than one, names are defined positionally, msb to lsb. If
  fewer signal names than <number of nodes> are specified, the <node> names are used for
  the remaining unspecified names.

Comments

The resulting file contains time and state values for the circuit nodes specified in the statement.
The file format is identical to that used by the digital file stimulus device (FSTIM). Thus, the
results of one simulation can be used to drive inputs of a subsequent simulation. See File
stimulus for more information on the file stimulus file format.

The optional parameters on the .VECTOR command can be used to control the file name,
column order, radix of the state values, and signal names which appear in the file header. Each
parameter is described in detail in the following table.

A different file name can be specified by using the FILE parameter. You can use multiple
.VECTOR commands to specify nodes for the same file.
**Commands**

.**WATCH** (watch analysis results)

**Purpose**
The .WATCH command/statement outputs results from DC, AC, and transient analyses to the PSpice display in text format while the simulation is running.

**General form**
```plaintext
.WATCH [DC][AC][TRAN] + [<output variable> [<lower limit value>,<upper limit value>]]*
```

**Examples**
- .WATCH DC V(3) (-1V,4V) V(2,3) V(R1)
- .WATCH AC VM(2) VP(2) VMC(Q1)
- .WATCH TRAN VBE(Q13) (0V,5V) ID(M2) I(VCC) (0,500mA)
- .WATCH DC V([RESET]) (2.5V,10V)

**Arguments and options**

DC, AC, and TRAN
The analysis types whose results are displayed during the simulation. You only need to specify one analysis type per .WATCH command, but there can be a .WATCH command for each analysis type in the circuit.

<output variable>
A maximum of eight output variables are allowed on a single .WATCH statement.

<lower limit value>,<upper limit value>
Specifies the normal operating range of that particular output variable. If the range is exceeded during the simulation, the simulator beeps and pauses. At this point, the simulation can be canceled or continued. If continued, the check for that output variable’s boundary condition is eliminated. Each output variable can have its own value range.

The first example displays three output variables on the screen. The first variable, V(3), has an operating range set from minus one volt to four volts. If during the simulation the voltage at node three exceeds four volts, the simulation will pause. If the simulation is allowed to proceed, and node three continues to rise in value, the simulation is then not interrupted. However, if the simulation is allowed to continue and V(3) falls below -1.0 volt, the simulation would again pause because a new boundary condition was exceeded.

Up to three output variables can be seen on the display at one time. More than three variables can be specified, but they are not all displayed.

The possible output variables are given in .**PROBE** (Probe), with the exception that digital nodes cannot be used and group delay is not available.
**.WCASE (sensitivity/worst-case analysis)**

**Purpose**
The .WCASE statement causes a sensitivity and worst-case analysis of the circuit to be performed.

**General form**
```
.WCASE <analysis> <output variable> <function> [option]*
```

**Examples**
```
.WCASE TRAN V(5) YMAX
.WCASE DC IC(Q7) YMAX VARY DEV
.WCASE AC VP(13,5) YMAX DEVICES RQ OUTPUT ALL
.WCASE TRAN V([OUT1],[OUT2]) YMAX RANGE(.4u,.6u)
+ LIST OUTPUT ALL VARY DEV HI
```

**Arguments and options**

- **<analysis>**
  Only one of DC, AC, or TRAN must be specified for <analysis>. This analysis is repeated in subsequent passes of the worst-case analysis. All requested analyses are performed during the nominal pass. Only the selected analysis is performed during subsequent passes.

- **<output variable>**
  Identical in format to that of a .PRINT (print) output variable.

- **<function>**
  Specifies the operation to be performed on the values of the <output variable> to reduce these to a single value. This value is the basis for the comparisons between the nominal and subsequent runs. The <function> must be one of the following:

<table>
<thead>
<tr>
<th>Function</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>YMAX</td>
<td>Find the absolute value of the greatest difference in each waveform from the nominal run.</td>
</tr>
<tr>
<td>MAX</td>
<td>Find the maximum value of each waveform.</td>
</tr>
<tr>
<td>MIN</td>
<td>Find the minimum value of each waveform.</td>
</tr>
<tr>
<td>RISE_EDGE(&lt;value&gt;)</td>
<td>Find the first occurrence of the waveform crossing above the threshold &lt;value&gt;. The waveform must have one or more points at or below &lt;value&gt; followed by one above; the output value listed is where the waveform increases above &lt;value&gt;.</td>
</tr>
<tr>
<td>FALL_EDGE(&lt;value&gt;)</td>
<td>Find the first occurrence of the waveform crossing below the threshold &lt;value&gt;. The waveform must have one or more points at or above &lt;value&gt; followed by one below; the output value listed is where the waveform decreases below &lt;value&gt;.</td>
</tr>
</tbody>
</table>
[option]*
Could have any number of the following.

<table>
<thead>
<tr>
<th>[option]</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST</td>
<td>Prints the updated model parameters for the sensitivity analysis. This does not affect the Probe data generated by the simulation.</td>
</tr>
<tr>
<td>OUTPUT ALL</td>
<td>Prints output from the sensitivity runs, after the nominal (first) run. The output from any run is governed by the .PRINT, .PLOT, and</td>
</tr>
<tr>
<td></td>
<td>.PROBE command in the file. If OUTPUT ALL is omitted, then only the nominal and worst-case runs produce output. OUTPUT ALL ensures that all sensitivity information is saved for Probe.</td>
</tr>
<tr>
<td>RANGE* (&lt;low value&gt;, &lt;high value&gt;)</td>
<td>Restricts the range over which &lt;function&gt; can be evaluated. An asterisk * can be used in place of a &lt;value&gt; to show for all values.</td>
</tr>
<tr>
<td></td>
<td>For example see the next two rows.</td>
</tr>
<tr>
<td>YMAX RANGE(*,5)</td>
<td>YMAX is evaluated for values of the sweep variable (e.g., time, and frequency) of .5 or less.</td>
</tr>
<tr>
<td>MAX RANGE(-1,*)</td>
<td>The maximum of the output variable is found for values of the sweep variable of -1 or more.</td>
</tr>
<tr>
<td>HI or LOW</td>
<td>Specify the direction which &lt;function&gt; should move for the worst-case run is to go (relative to the nominal). If &lt;function&gt; is</td>
</tr>
<tr>
<td></td>
<td>YMAX or MAX, the default is HI, otherwise the default is LOW.</td>
</tr>
<tr>
<td>VARY DEV</td>
<td>VARY LOT</td>
</tr>
<tr>
<td>DEVICES (list of device types)</td>
<td>By default, all devices are included in the sensitivity and worst-case analyses. The devices considered can be limited by listing the device types after the keyword DEVICES. Do not use any spaces or tabs in the devices list. For example, to only perform the analysis on resistors and MOSFETs, enter: DEVICES RM</td>
</tr>
</tbody>
</table>

* If RANGE is omitted, then <function> is evaluated over the whole sweep range. This is equivalent to RANGE(*, *).
| Commands | .WCASE (sensitivity/worst-case analysis) |
Multiple runs of the selected analysis (DC, AC, or transient) are performed while parameters are varied. Unlike .MC (Monte Carlo analysis), .WCASE varies only one parameter per run. This allows PSpice to calculate the sensitivity of the output waveform to each parameter. Once all the sensitivities are known, one final run is performed using all parameters varied so as to produce the worst-case waveform. The sensitivity and worst-case runs are performed using variations on model parameters as specified by the DEV and LOT tolerances on each .MODEL (model definition) parameter (see page 1-52 for details on the DEV and LOT tolerances). Other specifications on the .WCASE command control the output generated by the analysis.

You can run either .MC or .WCASE for a circuit, but not both in the same circuit.
A statement beginning with an asterisk * is a comment line, which PSpice ignores.

* [any text]

* This is an example of
  * a multiple-line comment

Use an asterisk at the beginning of each line you want to be a comment. A single asterisk does not extend to subsequent lines. For example:

* .MODEL ABC NMOS (. . . .
  + . . . .)

produces an error message, because the second line is not covered by the first asterisk.

The use of comment statements throughout the input is recommended. It is good practice to insert a comment line just before a subcircuit definition to identify the nodes, for example:

* +IN -IN V+ V- +OUT -OUT
  .SUBCKT OPAMP 100 101 1 2 200 201

or to identify major blocks of circuitry.
### ; (in-line comment)

**Purpose**
A semicolon ; is treated as the end of a line.

**General form**
circuit file text ;[any text]

**Examples**
R13 6 8 10 ; R13 is a feedback resistor
C3 15 0 .1U ; decouple supply

---

The simulator moves on to the next line in the circuit file. The text on the line after the semicolon ; is a comment and has no effect. The use of comments throughout the input is recommended. This type of comment can also replace comment lines, which must start with * in the first column.

Trailing in-line comments that extend to more than one line can use a semicolon to mark the beginning of the subsequent comment lines, as shown in the example.
+ (line continuation)

Purpose
A plus sign + is treated as the continuation of the previous line.

General form
`circuit file text`  
`+ more text`

Examples
```
.DISTRIBUTION bi_modal (-1.1) (-.5.1) (-.5.0) (.5.0)  
+ (.5.1) (1.1)
```

Comments
Because the simulator reads the line preceded by a plus sign as a continuation of the previous line, you can use the plus sign to break up long lines of command text.
Differences between PSpice and Berkeley SPICE2

The version of SPICE2 referred to is SPICE2G.6 from the University of California at Berkeley.

PSpice runs any circuit that SPICE2 can run, with these exceptions:

1. Circuits that use .DISTO (small-signal distortion) analysis. U.C. Berkeley SPICE supports the .DISTO analysis, but contains errors. Also, the special distortion output variables (e.g., HD2 and DIM3) are not available. Instead of the .DISTO analysis, MicroSim recommends running a transient analysis and looking at the output spectrum using the Fourier transform mode in Probe. This technique shows the distortion (spectral) products for both small-signal and large-signal distortion.

2. These options on the .OPTIONS (analysis options) statement are not available in PSpice:
   - LIMTIM: it is assumed to be 0.
   - LVLCOD: no in-line machine code is generated.
   - METHOD: a combination of trapezoidal and gear integration is always used.
   - MAXORD: a combination of trapezoidal and gear integration is always used.
   - LVLTIM: truncation error time step control is always used.
   - ITL3: truncation error time step control is always used.

3. The IN= option on the .WIDTH statement is not available. PSpice always reads the entire input file regardless of how long the input lines are.

4. Voltage coefficients for capacitors, and current coefficients for inductors must be put into a .MODEL (model definition) statement instead of on the device statement.

5. PSpice does not allow the use of nested subcircuit definitions.

   If this construct is used:
   .SUBCKT ABC 1 2 3
   ...
   .SUBCKT DEF 4 5 6
   ...
   .ENDS
   ...
   .ENDS

   It is recommended that the definitions be separated into:
   .SUBCKT ABC 1 2 3
   ...
   X1 ... DEF
   ...
   .ENDS
   .SUBCKT DEF 4 5 6
   ...
   .ENDS
You can nest subcircuit calls.

6 The .ALTER command is not supported in PSpice. Instead, use the \texttt{.STEP (parametric analysis)} command to modify specific parameters over multiple PSpice runs.

7 The syntax for the one-dimensional POLY form of E, F, G, and H (\textit{Voltage-controlled voltage source} and \textit{Current-controlled current source}) devices is different. PSpice requires a dimension specification of the form POLY(1), while SPICE does not.

PSpice produces basically the same results as SPICE. There can be some small differences, especially for values crossing zero, due to the corrections made for convergence problems.

The semiconductor device models are the same as in SPICE.
<table>
<thead>
<tr>
<th>Commands</th>
<th>Differences between PSpice and Berkeley SPICE2</th>
</tr>
</thead>
</table>

104
## Analog devices

<table>
<thead>
<tr>
<th>Letter</th>
<th>Device type</th>
<th>Letter</th>
<th>Device type</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>GaAsFET</td>
<td>N</td>
<td>Digital input (N device)</td>
</tr>
<tr>
<td>C</td>
<td>Capacitor</td>
<td>O</td>
<td>Digital output (O device)</td>
</tr>
<tr>
<td>D</td>
<td>Diode</td>
<td>Q</td>
<td>Bipolar transistor</td>
</tr>
<tr>
<td>E</td>
<td>Voltage-controlled voltage source</td>
<td>R</td>
<td>Resistor</td>
</tr>
<tr>
<td>F</td>
<td>Current-controlled current source</td>
<td>S</td>
<td>Voltage-controlled switch</td>
</tr>
<tr>
<td>G</td>
<td>Voltage-controlled current source</td>
<td>T</td>
<td>Transmission line</td>
</tr>
<tr>
<td>H</td>
<td>Current-controlled voltage source</td>
<td>U</td>
<td>Digital primitive summary</td>
</tr>
<tr>
<td>I</td>
<td>Independent current source &amp; stimulus</td>
<td>U STIM</td>
<td>Stimulus devices</td>
</tr>
<tr>
<td>J</td>
<td>Junction FET</td>
<td>V</td>
<td>Independent voltage source &amp; stimulus</td>
</tr>
<tr>
<td>K</td>
<td>Inductor coupling (and magnetic core)</td>
<td>W</td>
<td>Current-controlled switch</td>
</tr>
<tr>
<td>K</td>
<td>Transmission line coupling</td>
<td>X</td>
<td>Subcircuit instantiation</td>
</tr>
<tr>
<td>L</td>
<td>Inductor</td>
<td>Z</td>
<td>IGBT</td>
</tr>
<tr>
<td>M</td>
<td>MOSFET</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Analog devices

This chapter describes the different types of analog devices supported by PSpice and PSpice A/D. These device types include analog primitives, independent and controlled sources, and subcircuit calls. Each device type is described separately, and each description includes the following information as applicable:

- A description and an example of the proper netlist syntax.
- The corresponding model types and their description.
- The corresponding list of model parameters and their descriptions.
- The equivalent circuit diagram and characteristic equations for the model (as required).
- References to publications that the model is based on.

These analog devices include all of the standard circuit components that normally are not considered part of the two-state (binary) devices that are found in the digital devices.

The model library consists of analog models of off-the-shelf parts that you can use directly in your circuit designs. Refer to the online Library List for available device models and the libraries they are located in. You can also implement models using the .MODEL (model definition) statement and implement macromodels as subcircuits using the .SUBCKT (subcircuit) statement.

The Device types summary table lists all of the analog device primitives supported by PSpice A/D. Each primitive is described in detail in the sections following the table.
Analog devices

Device types

PSpice supports many types of analog devices, including sources and general subcircuits. PSpice A/D also supports digital devices. The supported devices are categorized into device types, each of which can have one or more model types. For example, the BJT device type has three model types: NPN, PNP, and LPNP (Lateral PNP). The description of each device type includes a description of any of the model types it supports.

The device declarations in the netlist always begin with the name of the individual device (instance). The first letter of the name determines the device type. What follows the name depends on the device type and its requested characteristics. Below is a summary of the device types and the general form of their declaration formats.

The table below includes the designator (letter) used in device modeling for each device type.

### Analog device summary

<table>
<thead>
<tr>
<th>Device type</th>
<th>Letter</th>
<th>Declaration format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bipolar transistor</td>
<td>Q</td>
<td>Q&lt;name&gt; &lt;collector node&gt; &lt;base node&gt; &lt;emitter node&gt; + [substrate node] &lt;model name&gt; [area value]</td>
</tr>
<tr>
<td>Capacitor</td>
<td>C</td>
<td>C&lt;name&gt; &lt;+ node&gt; &lt;- node&gt; [model name] &lt;value&gt; + [IC=&lt;initial value&gt;]</td>
</tr>
<tr>
<td>Voltage-controlled voltage source</td>
<td>E</td>
<td>E&lt;name&gt; &lt;+ node&gt; &lt;- node&gt; &lt;+ controlling node&gt; &lt;+ controlling node&gt; &lt;gain&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(additional Analog Behavioral Modeling forms: VALUE, TABLE, LAPLACE, FREQ, and CHEBYSHEV; additional POLY form)</td>
</tr>
<tr>
<td>Voltage-controlled current source</td>
<td>G</td>
<td>G&lt;name&gt; &lt;+ node&gt; &lt;- node&gt; &lt;+ controlling node&gt; &lt;+ controlling node&gt; &lt;transconductance&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(additional Analog Behavioral Modeling forms: VALUE, TABLE, LAPLACE, FREQ, and CHEBYSHEV; additional POLY form)</td>
</tr>
<tr>
<td>Current-controlled current source</td>
<td>F</td>
<td>F&lt;name&gt; &lt;+ node&gt; &lt;- node&gt; &lt;controlling V device name&gt; + &lt;gain&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(additional POLY form)</td>
</tr>
<tr>
<td>Current-controlled switch</td>
<td>W</td>
<td>W&lt;name&gt; &lt;+ switch node&gt; &lt;- switch node&gt; + &lt;controlling V device name&gt; &lt;model name&gt;</td>
</tr>
<tr>
<td>Current-controlled voltage source</td>
<td>H</td>
<td>H&lt;name&gt; &lt;+ node&gt; &lt;- node&gt; &lt;controlling V device name&gt; + &lt;transresistance&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(additional POLY form)</td>
</tr>
<tr>
<td>Digital input (N device)</td>
<td>N</td>
<td>N&lt;name&gt; &lt;interface node&gt; &lt;low level node&gt; &lt;high level node&gt; + &lt;model name&gt; &lt;input specification&gt;</td>
</tr>
</tbody>
</table>
## Analog device summary (continued)

<table>
<thead>
<tr>
<th>Device type</th>
<th>Letter</th>
<th>Declaration format</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Digital output (O Device)</strong></td>
<td>O</td>
<td>O&lt;name&gt; &lt;interface node&gt; &lt;low level node&gt; &lt;high level node&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ &lt;model name&gt; &lt;output specification&gt;</td>
</tr>
<tr>
<td><strong>Digital primitive summary</strong></td>
<td>U</td>
<td>U&lt;name&gt; &lt;primitive type&gt; (([parameter value]*)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ &lt;digital power node&gt; &lt;digital ground node&gt; &lt;node&gt;*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ &lt;timing model name&gt;</td>
</tr>
<tr>
<td><strong>Stimulus devices</strong></td>
<td>U STIM</td>
<td>U&lt;name&gt; STIM (&lt;width value&gt;, &lt;format value&gt;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ &lt;digital power node&gt; &lt;digital ground node&gt; &lt;node&gt;*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ &lt;I/O model name&gt; [TIMESTEP=&lt;stepsize value&gt;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ &lt;waveform description&gt;</td>
</tr>
<tr>
<td><strong>Diode</strong></td>
<td>D</td>
<td>D&lt;name&gt; &lt;anode node&gt; &lt;cathode node&gt; &lt;model name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[area value]</td>
</tr>
<tr>
<td><strong>GaAsFET</strong></td>
<td>B</td>
<td>B&lt;name&gt; &lt;drain node&gt; &lt;gate node&gt; &lt;source node&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ &lt;model name&gt; [area value]</td>
</tr>
<tr>
<td><strong>Independent current source &amp; stimulus</strong></td>
<td>I</td>
<td>I&lt;name&gt; &lt;- node&gt; &lt;- node&gt; [[DC] &lt;value&gt;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ [AC &lt;magnitude value&gt; [phase value]] [transient specification]</td>
</tr>
<tr>
<td><strong>Independent voltage source &amp; stimulus</strong></td>
<td>V</td>
<td>V&lt;name&gt; &lt;- node&gt; &lt;- node&gt; [[DC] &lt;value&gt;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ [AC &lt;magnitude value&gt; [phase value]] [transient specification]</td>
</tr>
<tr>
<td><strong>Inductor</strong></td>
<td>L</td>
<td>L&lt;name&gt; &lt;- node&gt; &lt;- node&gt; [model name] &lt;value&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ [IC=&lt;initial value&gt;]</td>
</tr>
<tr>
<td><strong>Inductor coupling (and magnetic core)</strong></td>
<td>K</td>
<td>K&lt;name&gt; L&lt;inductor name&gt; &lt;L&lt;inductor name&gt;&gt;*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ &lt;coupling value&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>K&lt;name&gt; &lt;L&lt;inductor name&gt;&gt;* &lt;coupling value&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ &lt;model name&gt; [size value]</td>
</tr>
<tr>
<td><strong>IGBT</strong></td>
<td>Z</td>
<td>Z&lt;name&gt; &lt;collector&gt; &lt;gate&gt; &lt;emitter&gt; &lt;model name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ [AREA=&lt;value&gt;] [WB=&lt;value&gt;] [AGD=&lt;value&gt;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ [KP=&lt;value&gt;] [TAU=&lt;value&gt;]</td>
</tr>
<tr>
<td><strong>Junction FET</strong></td>
<td>J</td>
<td>J&lt;name&gt; &lt;drain node&gt; &lt;gate node&gt; &lt;source node&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ &lt;model name&gt; [area value]</td>
</tr>
<tr>
<td><strong>MOSFET</strong></td>
<td>M</td>
<td>M&lt;name&gt; &lt;drain node&gt; &lt;gate node&gt; &lt;source node&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ [bulk/substrate node] &lt;model name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ [common model parameter]*</td>
</tr>
<tr>
<td><strong>Resistor</strong></td>
<td>R</td>
<td>R&lt;name&gt; &lt;- node&gt; &lt;- node&gt; [model name] &lt;value&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ [TC=&lt;linear temp. coefficient&gt;[,&lt;quadratic temp. coefficient]]</td>
</tr>
<tr>
<td><strong>Subcircuit instantiation</strong></td>
<td>X</td>
<td>X&lt;name&gt; [node]* &lt;subcircuit name&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ [PARAMS: &lt;name&gt;=&lt;value&gt;&gt;*] [TEXT: &lt;name&gt;=&lt;text value&gt;&gt;]</td>
</tr>
</tbody>
</table>
Analog devices

Analog device summary (continued)

<table>
<thead>
<tr>
<th>Device type</th>
<th>Letter</th>
<th>Declaration format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmission line</td>
<td>T</td>
<td>T&lt;name&gt; &lt;A port + node&gt; &lt;A port - node&gt; + &lt;B port + node&gt; &lt;B port - node&gt; &lt;ideal or lossy specification&gt;</td>
</tr>
<tr>
<td>Transmission line coupling</td>
<td>K</td>
<td>K&lt;name&gt; T&lt;line name&gt; &lt;T&lt;line name&gt;&gt;* + CM=&lt;coupling capacitance&gt; LM=&lt;coupling inductance&gt;</td>
</tr>
<tr>
<td>Voltage-controlled switch</td>
<td>S</td>
<td>S&lt;name&gt; &lt;+ switch node&gt; &lt;- switch node&gt; + &lt;+ controlling node&gt; &lt;- controlling node&gt; &lt;model name&gt;</td>
</tr>
</tbody>
</table>
GaAsFET

**General form**

B<name> <drain node> <gate node> <source node> <model name> [area value]

**Examples**

BIN 100 10 0 GFAST
B13 22 14 23 GNOM 2.0

**Model form**

.MODEL <model name> GASFET [model parameters]

**Description**

The GaAsFET is modeled as an intrinsic FET using an ohmic resistance (\(RD/area\)) in series with the drain, another ohmic resistance (\(RS/area\)) in series with the source, and another ohmic resistance (\(RG\)) in series with the gate.

![GaAsFET Circuit Diagram]

**Arguments and options**

[area value]

The relative device area. Its default value is 1.0.

**Comments**

The **LEVEL** model parameter selects among different models for the intrinsic GaAsFET as follows:

- **LEVEL=1**  “Curtice” model (see reference [1])
- **LEVEL=2**  “Raytheon” or “Statz” model (see reference [3]), equivalent to the GaAsFET model in SPICE3
- **LEVEL=3**  “TOM” model by TriQuint (see reference [4])
- **LEVEL=4**  “Parker-Skellern” model (see reference [5] and [6])
- **LEVEL=5**  “TOM-2” model by TriQuint (see reference [7])

For more information, see References.
The TOM-2 model is based on the original TriQuint TOM model, retaining the desirable features of the TOM model, while improving accuracy in the subthreshold near cutoff and knee regions (Vds of 1 volt or less). This model includes additional temperature coefficients related to the drain current and corrects the major deficiencies in the behavior of the capacitance as a function of temperature.

**Capture parts**

The following table lists the set of GaAsFET breakout parts designed for customizing model parameters for simulation. These are useful for setting up Monte Carlo and worst-case analyses with device and/or lot tolerances specified for individual model parameters.

<table>
<thead>
<tr>
<th>part name</th>
<th>Model type</th>
<th>Property</th>
<th>Property description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BBREAK</td>
<td>GASFET</td>
<td>AREA</td>
<td>area scaling factor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MODEL</td>
<td>GASFET model name</td>
</tr>
</tbody>
</table>

**Setting operating temperature**

Operating temperature can be set to be different from the global circuit temperature by defining one of the model parameters: T_ABS, T_REL_GLOBAL, or T_REL_LOCAL. Additionally, model parameters can be assigned unique measurement temperatures using the T_MEASURED model parameter.
## Model parameters

### GaAsFET model parameters for all levels

<table>
<thead>
<tr>
<th>Model parameter</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF</td>
<td>flicker noise exponent</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>BETA</td>
<td>transconductance coefficient</td>
<td>amp/volt^2</td>
<td>0.1</td>
</tr>
<tr>
<td>BETATCE</td>
<td>BETA exponential temperature coefficient</td>
<td>%/°C</td>
<td>0</td>
</tr>
<tr>
<td>CDS</td>
<td>drain-source capacitance</td>
<td>farad</td>
<td>0</td>
</tr>
<tr>
<td>CGD</td>
<td>zero-bias gate-drain p-n capacitance</td>
<td>farad</td>
<td>0</td>
</tr>
<tr>
<td>CGS</td>
<td>zero-bias gate-source p-n capacitance</td>
<td>farad</td>
<td>0</td>
</tr>
<tr>
<td>EG</td>
<td>band gap voltage (barrier height)</td>
<td>eV</td>
<td>1.11</td>
</tr>
<tr>
<td>FC</td>
<td>forward-bias depletion capacitance coefficient</td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>IS</td>
<td>gate p-n saturation current</td>
<td>amp</td>
<td>1E-14</td>
</tr>
<tr>
<td>KF</td>
<td>flicker noise coefficient</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>LEVEL</td>
<td>model index (1, 2, 3, 4, or 5)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>N</td>
<td>gate p-n emission coefficient</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>RD</td>
<td>drain ohmic resistance</td>
<td>ohm</td>
<td>0</td>
</tr>
<tr>
<td>RG</td>
<td>gate ohmic resistance</td>
<td>ohm</td>
<td>0</td>
</tr>
<tr>
<td>RS</td>
<td>source ohmic resistance</td>
<td>ohm</td>
<td>0</td>
</tr>
<tr>
<td>TRD1</td>
<td>RD temperature coefficient (linear)</td>
<td>°C^-1</td>
<td>0</td>
</tr>
<tr>
<td>TRG1</td>
<td>RG temperature coefficient (linear)</td>
<td>°C^-1</td>
<td>0</td>
</tr>
<tr>
<td>TRS1</td>
<td>RS temperature coefficient (linear)</td>
<td>°C^-1</td>
<td>0</td>
</tr>
<tr>
<td>T_ABS</td>
<td>absolute temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_MEASURED</td>
<td>measured temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_REL_GLOBAL</td>
<td>relative to current temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_REL_LOCAL</td>
<td>relative to AKO model temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>VBI</td>
<td>gate p-n potential</td>
<td>volt</td>
<td>1.0</td>
</tr>
<tr>
<td>VTO</td>
<td>pinchoff voltage</td>
<td>volt</td>
<td>-2.5</td>
</tr>
<tr>
<td>VTOTC</td>
<td>VTO temperature coefficient</td>
<td>volt/°C</td>
<td>0</td>
</tr>
<tr>
<td>XTI</td>
<td>IS temperature exponent</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

* For information on T_ABS, T_MEASURED, T_REL_GLOBAL, and T_REL_LOCAL, see the `MODEL (model definition)` statement.
## GaAsFET model parameters specific to model levels

<table>
<thead>
<tr>
<th>Model parameter</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>level 1</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALPHA</td>
<td>saturation voltage parameter</td>
<td>volt⁻¹</td>
<td>2.0</td>
</tr>
<tr>
<td>LAMBDA</td>
<td>channel-length modulation</td>
<td>volt⁻¹</td>
<td>0</td>
</tr>
<tr>
<td>M</td>
<td>gate p-n grading coefficient</td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>TAU</td>
<td>conduction current delay time</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td><strong>level 2</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALPHA</td>
<td>saturation voltage parameter</td>
<td>volt⁻¹</td>
<td>2.0</td>
</tr>
<tr>
<td>B</td>
<td>doping tail extending parameter</td>
<td>volt⁻¹</td>
<td>0.3</td>
</tr>
<tr>
<td>LAMBDA</td>
<td>channel-length modulation</td>
<td>volt⁻¹</td>
<td>0</td>
</tr>
<tr>
<td>M</td>
<td>gate p-n grading coefficient</td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>TAU</td>
<td>conduction current delay time</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>VDELTA</td>
<td>capacitance transition voltage</td>
<td>volt</td>
<td>0.2</td>
</tr>
<tr>
<td>VMAX</td>
<td>capacitance limiting voltage</td>
<td>volt</td>
<td>0.5</td>
</tr>
<tr>
<td><strong>level 3</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALPHA</td>
<td>saturation voltage parameter</td>
<td>volt⁻¹</td>
<td>2.0</td>
</tr>
<tr>
<td>BTRK</td>
<td>auxiliary parameter for Monte Carlo analysis*</td>
<td>amp/volt³</td>
<td>0</td>
</tr>
<tr>
<td>DELTA</td>
<td>output feedback parameter</td>
<td>(amp-volt)⁻¹</td>
<td>0</td>
</tr>
<tr>
<td>DVT</td>
<td>auxiliary parameter for Monte Carlo analysis*</td>
<td>volt</td>
<td>0</td>
</tr>
<tr>
<td>DVTT</td>
<td>auxiliary parameter for Monte Carlo analysis*</td>
<td>volt</td>
<td>0</td>
</tr>
</tbody>
</table>
### GaAsFET model parameters specific to model levels (continued)

<table>
<thead>
<tr>
<th>Model parameter</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAMMA</td>
<td>static feedback parameter</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>M</td>
<td>gate p-n grading coefficient</td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>Q</td>
<td>power-law parameter</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>TAU</td>
<td>conduction current delay time</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>VDELT A</td>
<td>capacitance transition voltage</td>
<td>volt</td>
<td>0.2</td>
</tr>
<tr>
<td>VMAX</td>
<td>gate diode capacitance limiting voltage</td>
<td>volt</td>
<td>0.5</td>
</tr>
<tr>
<td>ACGAM</td>
<td>capacitance modulation</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>DELTA</td>
<td>output feedback parameter</td>
<td>(amp-volt)$^{-1}$</td>
<td>0</td>
</tr>
<tr>
<td>HFETA</td>
<td>high-frequency VGS feedback parameter</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>HFE1</td>
<td><strong>HFGAM</strong> modulation by VGD</td>
<td>volt$^{-1}$</td>
<td>0</td>
</tr>
<tr>
<td>HFE2</td>
<td><strong>HFGAM</strong> modulation by VGS</td>
<td>volt$^{-1}$</td>
<td>0</td>
</tr>
<tr>
<td>HFGAM</td>
<td>high-frequency VGD feedback parameter</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>HFG1</td>
<td><strong>HFGAM</strong> modulation by VSG</td>
<td>volt$^{-1}$</td>
<td>0</td>
</tr>
<tr>
<td>HFG2</td>
<td><strong>HFGAM</strong> modulation by VDG</td>
<td>volt$^{-1}$</td>
<td>0</td>
</tr>
<tr>
<td>IBD</td>
<td>gate junction breakdown current</td>
<td>amp</td>
<td>0</td>
</tr>
<tr>
<td>LAMBDA</td>
<td>channel-length modulation</td>
<td>volt$^{-1}$</td>
<td>0</td>
</tr>
<tr>
<td>LFGAM</td>
<td>low-frequency feedback parameter</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>LFG1</td>
<td><strong>LFGAM</strong> modulation by VSG</td>
<td>volt$^{-1}$</td>
<td>0</td>
</tr>
<tr>
<td>LFG2</td>
<td><strong>LFGAM</strong> modulation by VDG</td>
<td>volt$^{-1}$</td>
<td>0</td>
</tr>
<tr>
<td>MVST</td>
<td>subthreshold modulation</td>
<td>volt$^{-1}$</td>
<td>0</td>
</tr>
<tr>
<td>MXI</td>
<td>saturation knee-potential modulation</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>P</td>
<td>linear-region power law exponent</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Q</td>
<td>power-law parameter</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>TAU D</td>
<td>relaxation time for thermal reduction</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TAU G</td>
<td>relaxation time for GAM feedback</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>VBD</td>
<td>gate junction breakdown potential</td>
<td>volt</td>
<td>1</td>
</tr>
</tbody>
</table>
## GaAsFET model parameters specific to model levels (continued)

<table>
<thead>
<tr>
<th>Model parameter</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>VST</td>
<td>subthreshold potential</td>
<td>volt</td>
<td>0</td>
</tr>
<tr>
<td>XC</td>
<td>capacitance pinchoff reduction factor</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>XI</td>
<td>saturation knee potential factor</td>
<td></td>
<td>1000</td>
</tr>
<tr>
<td>Z</td>
<td>knee transition parameter</td>
<td></td>
<td>0.5</td>
</tr>
</tbody>
</table>

### level 5

<table>
<thead>
<tr>
<th>Model parameter</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPHA</td>
<td>saturation voltage parameter</td>
<td>volt⁻¹</td>
<td>2.0</td>
</tr>
<tr>
<td>ALPHATCE</td>
<td>$\text{ALPHA}$ temperature coefficient</td>
<td>%/°C</td>
<td>0</td>
</tr>
<tr>
<td>BTRK</td>
<td>auxiliary parameter for Monte Carlo analysis*</td>
<td>amp/volt³</td>
<td>0</td>
</tr>
<tr>
<td>CGDTCE</td>
<td>$\text{CGD}$ temperature coefficient</td>
<td>°C⁻¹</td>
<td>0</td>
</tr>
<tr>
<td>CGSTCE</td>
<td>$\text{CGS}$ temperature coefficient</td>
<td>°C⁻¹</td>
<td>0</td>
</tr>
<tr>
<td>DELTA</td>
<td>output feedback parameter</td>
<td>(amp·volt)⁻¹</td>
<td>0</td>
</tr>
<tr>
<td>DVT</td>
<td>auxiliary parameter for Monte Carlo analysis*</td>
<td>volt</td>
<td>0</td>
</tr>
<tr>
<td>DVTT</td>
<td>auxiliary parameter for Monte Carlo analysis*</td>
<td>volt</td>
<td>0</td>
</tr>
<tr>
<td>GAMMA</td>
<td>static feedback parameter</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>GAMMATCE</td>
<td>$\text{GAMMA}$ temperature coefficient</td>
<td>°C⁻¹</td>
<td>0</td>
</tr>
<tr>
<td>ND</td>
<td>subthreshold slope drain pull parameter</td>
<td>volt⁻¹</td>
<td>0</td>
</tr>
<tr>
<td>NG</td>
<td>subthreshold slope gate parameter</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Q</td>
<td>power-law parameter</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>TAU</td>
<td>conduction current delay time</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>VBITC</td>
<td>$\text{VBI}$ temperature coefficient</td>
<td>volt/°C</td>
<td>0</td>
</tr>
<tr>
<td>VDELTA</td>
<td>capacitance transition voltage</td>
<td>volt</td>
<td>0.2</td>
</tr>
<tr>
<td>VMAX</td>
<td>gate diode capacitance limiting voltage</td>
<td>volt</td>
<td>0.5</td>
</tr>
</tbody>
</table>

*See auxiliary model parameters BTRK, DVT, and DVTT.*
Auxiliary model parameters BTRK, DVT, and DVTT

The parameters BTRK, DVT, and DVTT are auxiliary model parameters that are used to make the Monte Carlo analysis easier when using PSpice. In the analysis, these affect the parameters VTO and BETA as follows:

\[
\begin{align*}
VTO &= VTO + DVT + DVTT \\
BETA &= BETA + BTRK \cdot (DVT + DVTT)
\end{align*}
\]

In Monte Carlo analysis, DEV tolerances placed on the DVT or DVTT cause variations in both VTO and BETA. PSpice does not support correlated DEV variations in Monte Carlo analysis. Without DVT and DVTT, DEV tolerances placed on VTO and BETA can result in independent variations; there is a definite correlation between VTO and BETA on real devices.

The BTRK, DVT, and DVTT parameters are also used to provide tracking between distinct GaAsFETs, such as between depletion mode and enhancement mode. PSpice already provides a limited mechanism for this, but only allows one DEV and one LOT (or LOT/n and DEV/n) tolerance per model parameter. The added parameters circumvent this restriction by extending the capability of Monte Carlo to model correlation between the critical model parameters.
GaAsFET equations

The equations in this section describe an N-channel GaAsFET. The following variables are used:

- $V_{gs}$ = intrinsic gate-intrinsic source voltage
- $V_{gd}$ = intrinsic gate-intrinsic drain voltage
- $V_{ds}$ = intrinsic drain-intrinsic source voltage
- $C_{ds}$ = drain-source capacitance
- $C_{gs}$ = gate-source capacitance
- $C_{gd}$ = gate-drain capacitance
- $V_{t} = \frac{k}{q} \cdot T$ (thermal voltage)
- $k = \text{Boltzmann constant}$
- $q = \text{electron charge}$
- $T = \text{analysis temperature (°K)}$

$T_{\text{nom}} = \text{nominal temperature (set by using } \texttt{OPTIONS (analysis options) T\text{NOM}=})$

Positive current is current flowing into a terminal (for example, positive drain current flows from the drain through the channel to the source).

GaAsFET equations for DC current: all levels

- $I_{g} = \text{gate current} = area \cdot (I_{gs} + I_{gd})$
- $I_{d} = \text{drain current} = area \cdot (I_{\text{drain}} - I_{gd})$
- $I_{s} = \text{source current} = area \cdot (-I_{\text{drain}} - I_{gs})$

where

- $I_{gs} = \text{gate-source leakage current}$
- $I_{gd} = \text{gate-drain leakage current}$
GaAsFET equations for DC current: specific to model levels

levels 1, 2, 3, and 5

\[ I_{gs} = I_{S} \cdot \left( e^{\frac{V_{gs}}{N \cdot V_{t}}} - 1 \right) \]

\[ I_{gd} = I_{S} \cdot \left( e^{\frac{V_{gd}}{N \cdot V_{t}}} - 1 \right) \]

**level 4**

\[ I_{gs} = I_{gsf} + I_{gsr} \]

where

\[ I_{gsf} = I_{S} \cdot \left( \frac{V_{gs}}{N \cdot V_{t}} - 1 \right) + V_{gs} \cdot G_{MIN} \]

and

\[ I_{gsr} = I_{BD} \cdot \left( \frac{V_{gs}}{1 - e^{V_{BD}}} \right) \]

\[ I_{gd} = I_{gdf} + I_{gdr} \]

where

\[ I_{gdf} = I_{S} \cdot \left( \frac{V_{gd}}{N \cdot V_{t}} - 1 \right) + V_{gd} \cdot G_{MIN} \]

and

\[ I_{gdr} = I_{BD} \cdot \left( \frac{V_{gd}}{1 - e^{V_{BD}}} \right) \]

**level 1: Idrain**

**Normal mode: Vds ≥ 0**

**Case 1**

for cutoff region: \( V_{gs} - V_{TO} < 0 \)

then: \( I_{drain} = 0 \)

**Case 2**

for linear & saturation region: \( V_{gs} - V_{TO} ≥ 0 \)

then: \( I_{drain} = BETA \cdot (1 + \Lambda BDA \cdot V_{ds}) \cdot (V_{gs} - V_{TO})^{2} \cdot tanh(ALPHA \cdot V_{ds}) \)

**Inverted mode: Vds < 0**

Switch the source and drain in the Normal mode equations.
GaAsFET equations for DC current: specific to model levels

level 2: Idrain

Normal mode: \( V_{ds} \geq 0 \)

Case 1

for cutoff region: \( V_{gs} - V_{TO} < 0 \)

then: \( I_{drain} = 0 \)

Case 2

for linear & saturation region: \( V_{gs} - V_{TO} \geq 0 \)

then: \( I_{drain} = \beta \cdot (1 + \lambda \cdot V_{ds}) \cdot (V_{gs} - V_{TO})^2 \cdot K_t / (1 + B \cdot (V_{gs} - V_{TO})) \)

where

\( K_t \) is a polynomial approximation of \( \tanh \).

for linear region:

\( 0 < V_{ds} < 3/\alpha \)

then:

\( K_t = 1 - \left( 1 - V_{ds} \cdot \alpha / 3 \right)^3 \)

for saturation region:

\( V_{ds} \geq 3/\alpha \)

then:

\( K_t = 1 \)

Inverted mode: \( V_{ds} < 0 \)

Switch the source and drain in the Normal mode equations.
GaAsFET equations for DC current: specific to model levels

level 3: Idrain

Normal mode: Vds ≥ 0

Case 1

for cutoff region:
Vgs - V_{to} < 0

then:
Idrain = 0

Case 2

for linear & saturation region:
Vgs - V_{to} ≥ 0

then:
Idrain = Idso/(1 + \Delta \cdot Vds \cdot Idso)

where
Idso = BETA \cdot (Vgs - V_{to}) \cdot Kt

and
V_{to} = VTO - GAMMA \cdot Vds

where
Kt is the same as for Level 2.

Inverted mode: Vds < 0

Switch the source and drain in the Normal Mode equations.

level 4: Idrain

Normal mode: Vds ≥ 0

\[
Idrain = \frac{Ids}{1 + \Delta \cdot p_{avg}}
\]

\[
V_{gs} = Vgs - VTO - \gamma_{lf} \cdot Vg_{d_{avg}} - \gamma_{hf} \cdot (Vgd - Vg_{d_{avg}}) - \eta_{hf} \cdot (Vgs - Vg_{s_{avg}})
\]

V_{ds} = Vds
Analog devices

GaAsFET equations for DC current: specific to model levels

Inverted mode: \( V_{ds} < 0 \)

\[
\begin{align*}
\text{Idrain} & = \frac{-\text{Ids}}{1 + \text{DELTA} \cdot P_{\text{avg}}} \\
V_{gst} & = V_{gd} - V_{TO} - \gamma_{lf} \cdot V_{gd_{avg}} - \gamma_{hf} \cdot (V_{gs} - V_{gd_{avg}}) - \eta_{hf} \cdot (V_{gd} - V_{gs_{avg}}) \\
V_{dst} & = -V_{ds}
\end{align*}
\]

where

\[
\begin{align*}
\text{Ids} & = \text{BETA} \cdot (1 - \text{LAMBDA} \cdot V_{dst}) \cdot (V_{gt} - (V_{gt} - V_{dt})^2) \\
P_{\text{avg}} & = V_{ds} \cdot \text{Ids} - TAUD \cdot \frac{d}{dt} P_{\text{avg}} \\
\gamma_{lf} & = \text{LFGAM} - \text{LFG1} \cdot V_{gs_{avg}} - \text{LFG2} \cdot V_{gd_{avg}} \\
V_{gd_{avg}} & = V_{gd} - TAUG \cdot \frac{d}{dt} V_{gd_{avg}} \quad \text{if: } V_{gd} \leq V_{gs} \\
& = V_{gs} - TAUG \cdot \frac{d}{dt} V_{gd_{avg}} \quad \text{if: } V_{gs} \times V_{gd} \\
\gamma_{hf} & = \text{HFGAM} - \text{HFG1} \cdot V_{gs_{avg}} - \text{HFG2} \cdot V_{gd_{avg}} \\
\eta_{hf} & = \text{HFETA} + \text{HFE1} \cdot V_{gd_{avg}} + \text{HFE2} \cdot V_{gs_{avg}} \\
V_{gs_{avg}} & = V_{gs} - TAUG \cdot \frac{d}{dt} V_{gs_{avg}} \quad \text{if: } V_{gd} \leq V_{gs} \\
& = V_{gd} - TAUG \cdot \frac{d}{dt} V_{gs_{avg}} \quad \text{if: } V_{gs} \times V_{gd} \\
V_{gt} & = V_{St} \cdot (1 + M_{VSt} \cdot V_{dst}) \cdot \ln \left( \exp \left( \frac{V_{gst}}{V_{St} \cdot (1 + M_{VSt} \cdot V_{dst})} \right) + 1 \right) \\
V_{dt} & = \frac{1}{2} \cdot \sqrt{D_{dp} \cdot \sqrt{1 + Z + V_{sat}}}^2 + Z \cdot V_{sat}^2 - \frac{1}{2} \cdot \sqrt{D_{dp} \cdot \sqrt{1 + Z - V_{sat}}}^2 + Z \cdot V_{sat}^2 \\
V_{dp} & = V_{dst} \cdot P \cdot \left( \frac{V_{gt}}{V_{Bt} - V_{TO}} \right)^{P - Q} \\
V_{sat} & = \frac{V_{gt} \cdot (V_{gt} \cdot M_{XI} + XI \cdot (V_{Bt} - V_{TO}))}{V_{gt} + V_{gt} \cdot M_{XI} + XI \cdot (V_{Bt} - V_{TO})}
\end{align*}
\]
GaAsFET equations for DC current: specific to model levels

level 5: Idrain

Normal mode: Vds ≥ 0

Case 1

For cutoff region:
\[
V_{gs} - V_{TO} + \text{GAMMA} \cdot V_{ds} \leq 0 \text{ AND } NG + ND \cdot V_{ds} = 0
\]

then:
\[
Idrain = 0
\]

Case 2

For linear and saturation region:
\[
V_{gs} - V_{TO} + \text{GAMMA} \cdot V_{ds} > 0 \text{ OR } NG + ND \cdot V_{ds} \neq 0
\]

then:
\[
Idrain = \frac{Idso}{1 + \delta \cdot V_{ds} \cdot Idso}
\]

where
\[
Idso = \text{BETA} \cdot (V_{g})^Q \cdot \frac{\text{ALPHA} \cdot V_{ds}}{\sqrt{1 + (\text{ALPHA} \cdot V_{ds})^2}}
\]

\[
V_{g} = Q \cdot V_{st} \cdot \log\left(\exp\left(\frac{V_{gs} - (V_{TO} + \text{GAMMA} \cdot V_{ds})}{Q \cdot V_{st}}\right) + 1\right)
\]

\[
V_{st} = (NG + ND \cdot V_{ds}) \cdot \left(\frac{kT}{q}\right)
\]

Inverted mode: Vds < 0

Switch the source and drain in the Normal mode equations.
GaAsFET equations for capacitance

All capacitances are between terminals of the intrinsic GaAsFET (i.e., to the inside of the ohmic drain, source, and gate resistances).

<table>
<thead>
<tr>
<th>all levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>For all conditions: $C_{ds} = area \cdot C_{DS}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>level 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>For: $V_{gs} \leq FC \cdot V_{BI}$ $C_{gs} = area \cdot C_{GS} \cdot (1 - V_{gs}/V_{BI})^M$</td>
</tr>
<tr>
<td>For: $V_{gs} &gt; FC \cdot V_{BI}$ $C_{gs} = area \cdot C_{GS} \cdot (1 - FC \cdot (1+M+M \cdot V_{gs}/V_{BI})$</td>
</tr>
<tr>
<td>For: $V_{gd} \leq FC \cdot V_{BI}$ $C_{gd} = area \cdot C_{GD} \cdot (1 - V_{gd}/V_{BI})^M$</td>
</tr>
<tr>
<td>For: $V_{gd} &gt; FC \cdot V_{BI}$ $C_{gd} = area \cdot C_{GD} \cdot (1 - FC \cdot (1+M+M \cdot V_{gd}/V_{BI})$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>levels 2, 3, and 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{gs} = area \cdot (C_{GS} \cdot K_{2} \cdot K_{1}/(1 - V_{n}/V_{BI})^{1/2} + C_{GD} \cdot K_{3})$</td>
</tr>
<tr>
<td>$C_{gd} = area \cdot (C_{GS} \cdot K_{3} \cdot K_{1}/(1 - V_{n}/V_{BI})^{1/2} + C_{GD} \cdot K_{2})$</td>
</tr>
</tbody>
</table>

where:

- $K_{1} = (1 + (V_{e} - V_{TO})^2 + V_{DELTA2}^{1/2})/2$
- $K_{2} = (1 + (V_{gs} - V_{gd})^2 + (1/ALPHA)^2)^{1/2}/2$
- $K_{3} = (1 - (V_{gs} - V_{gd})^2 + (1/ALPHA)^2)^{1/2}/2$
- $V_{e} = (V_{gs} + V_{gd} + ((V_{gs} - V_{gd})^2 + (1/ALPHA)^2)^{1/2})/2$

if: $(V_{e} + V_{TO} + ((V_{e} - V_{TO})^2 + V_{DELTA2}^{1/2})/2 < V_{MAX}$

then: $V_{n} = (V_{e} + V_{TO} + ((V_{e} - V_{TO})^2 + V_{DELTA2}^{1/2})/2$
else: $V_{n} = V_{MAX}$
Charge storage is implemented using a modified Statz model.

\[
C_{gs} = \frac{1}{2} \cdot K_1 \cdot \left( 1 - 2ACGAM \cdot \frac{V_{ds}}{\sqrt{V_{ds}^2 + \alpha^2}} \right) + \frac{1}{2} \cdot CGD \cdot \text{area} \cdot \left( 1 + 2ACGAM \cdot \frac{V_{ds}}{\sqrt{V_{ds}^2 + \alpha^2}} \right)
\]

\[
C_{gd} = \frac{1}{2} \cdot K_1 \cdot \left( 1 - 2ACGAM \cdot \frac{V_{ds}}{\sqrt{V_{ds}^2 + \alpha^2}} \right) + \frac{1}{2} \cdot CGD \cdot \text{area} \cdot \left( 1 - 2ACGAM + \frac{V_{ds}}{\sqrt{V_{ds}^2 + \alpha^2}} \right)
\]

where:

\[
K_1 = \frac{1}{2} \cdot \frac{CGS}{\sqrt{1 - V_{ge}/VBI}} \left[ 1 + XC + (1 - XC) \cdot \frac{V_{gb}}{V_{gb}^2 + 0.2^2} \right]
\]

if: \( V_x < FC \cdot VBI \) then: \( V_{ge} = V_x \)

if: \( V_x \geq FC \cdot VBI \) then: \( V_{ge} = VBI \left[ - \frac{4(1 - FC)^3}{\left( 2 - 3FC + \frac{V_x}{VBI} \right)^2} \right] \)

\[
V_x = V_{gs} + ACGAM \cdot V_{ds} - \frac{1}{2} \left( V_{gn} - \sqrt{V_{gn}^2 + 0.2^2} \right) - \frac{1}{2} \left( V_{gn} - \sqrt{V_{gn}^2 + 0.2^2} \right)
\]

\[
V_{gn} = \left( (V_{gs} + ACGAM) \cdot V_{ds} - VTO \right) - \frac{1}{2} \left( V_{ds} - \sqrt{V_{ds}^2 + \alpha^2} \right) \cdot (1 - XC)
\]

where:

\[
\alpha = \frac{XI}{XI + 1} \cdot \frac{VBI - VTO}{2}
\]

If the source and drain potentials swap, the model reverses over a range set by \( \alpha \). The model maintains a straight line relation between gate-source capacitance and gate bias in the region \( Vgs > FC \cdot VBI \).
## GaAsFET equations for temperature effect

### all levels

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{TO}(T) = V_{TO} + V_{TOTC} \cdot (T - T_{nom}) )</td>
<td>Voltage offset for temperature effect</td>
</tr>
<tr>
<td>( BETA(T) = BETA \cdot 1.01^{BETATCE \cdot (T - T_{nom})} )</td>
<td>Temperature coefficient for Beta</td>
</tr>
<tr>
<td>( IS(T) = IS \cdot e^{(T/T_{nom} - 1) \cdot \frac{E_G(T_{nom}) \cdot T}{N \cdot V_t} \cdot \frac{T}{T_{nom}}} )</td>
<td>Source current expression</td>
</tr>
<tr>
<td>( RG(T) = RG \cdot (1 + TRG1 \cdot (T - T_{nom})) )</td>
<td>Resistance expression</td>
</tr>
<tr>
<td>( RD(T) = RD \cdot (1 + TRD1 \cdot (T - T_{nom})) )</td>
<td>Resistance expression</td>
</tr>
<tr>
<td>( RS(T) = RS \cdot (1 + TRS1 \cdot (T - T_{nom})) )</td>
<td>Resistance expression</td>
</tr>
<tr>
<td>( CGS(T) = CGS \cdot (1 + M \cdot (0.0004 \cdot (T - T_{nom}) + (1 - VBI(T)/VBI))) )</td>
<td>Conductance expression</td>
</tr>
<tr>
<td>( CGD(T) = CGD \cdot (1 + M \cdot (0.0004 \cdot (T - T_{nom}) + (1 - VBI(T)/VBI))) )</td>
<td>Conductance expression</td>
</tr>
<tr>
<td>( VBI(T) = VBI \cdot \frac{T}{T_{nom}} - 3 \cdot Vt \cdot \ln \left( \frac{T}{T_{nom}} \right) - \frac{E_G(T_{nom}) \cdot T}{T_{nom}} + E_G(T) )</td>
<td>Base-emitter voltage expression</td>
</tr>
</tbody>
</table>

where:

- \( E_G(T) = \) silicon bandgap energy = 1.16 - 0.000702 \cdot (T+1108) \)

### levels 1, 2, 3, and 4

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( CGS(T) = CGS \cdot (1 + M \cdot (0.0004 \cdot (T - T_{nom}) + (1 - VBI(T)/VBI))) )</td>
<td>Conductance expression</td>
</tr>
<tr>
<td>( CGD(T) = CGD \cdot (1 + M \cdot (0.0004 \cdot (T - T_{nom}) + (1 - VBI(T)/VBI))) )</td>
<td>Conductance expression</td>
</tr>
<tr>
<td>( VBI(T) = VBI \cdot \frac{T}{T_{nom}} - 3 \cdot Vt \cdot \ln \left( \frac{T}{T_{nom}} \right) - E_G(T_{nom}) \cdot \frac{T}{T_{nom}} + E_G(T) )</td>
<td>Base-emitter voltage expression</td>
</tr>
</tbody>
</table>

### level 5

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( ALPHA(T) = ALPHA \cdot 1.01^{ALPHATCE \cdot (T - T_{nom})} )</td>
<td>Temperature coefficient for Alpha</td>
</tr>
<tr>
<td>( GAMMA(T) = GAMMA + GAMMATC \cdot (T - T_{nom}) )</td>
<td>Temperature coefficient for Gamma</td>
</tr>
<tr>
<td>( VBI(T) = VBI + VBITC \cdot (T - T_{nom}) )</td>
<td>Base-emitter voltage expression</td>
</tr>
<tr>
<td>( VMAX(T) = VMAX + VBITC \cdot (T - T_{nom}) )</td>
<td>Maximum voltage expression</td>
</tr>
<tr>
<td>( CGS(T) = CGS \cdot (1 + CGSTCE \cdot (T - T_{nom})) )</td>
<td>Conductance expression</td>
</tr>
<tr>
<td>( CGD(T) = CGD \cdot (1 + CGDTCE \cdot (T - T_{nom})) )</td>
<td>Conductance expression</td>
</tr>
</tbody>
</table>
GaAsFET equations for noise

Noise is calculated assuming a 1.0-hertz bandwidth, using the following spectral power densities (per unit bandwidth).

<table>
<thead>
<tr>
<th>parasitic resistance thermal noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_s = 4 \cdot k \cdot T / (R_s/\text{area})$</td>
</tr>
<tr>
<td>$I_d = 4 \cdot k \cdot T / (R_d/\text{area})$</td>
</tr>
<tr>
<td>$I_g = 4 \cdot k \cdot T / R_g$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>intrinsic GaAsFET shot and flicker noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_d = 4 \cdot k \cdot T \cdot \text{gm} \cdot 2/3 + K_F \cdot I_d^{AF} / \text{FREQUENCY}$</td>
</tr>
</tbody>
</table>

where:

\[
gm = \frac{dI_{\text{drain}}}{dV_{\text{gs}}} \text{ (at the DC bias point)}
\]
References

For more information on this GaAsFET model, refer to:


Capacitor

General form  
\[ C\text{name} \ (\text{(+)} \text{ node}) \ (\text{(-)} \text{ node}) \ [\text{model name}] \ [\text{value}] \ [\text{IC}=<\text{initial value}>] \]

Examples  
\[ \text{CLOAD 15 0 20pF} \]
\[ \text{C2 1 2 .2E-12 IC=1.5V} \]
\[ \text{CFDBCK 3 33 CMOD 10pF} \]

Model form  
\[ .\text{MODEL} \ <\text{model name}> \ \text{CAP} \ [\text{model parameters}] \]

Arguments and options  

(+ and (-) nodes  
Define the polarity when the capacitor has a positive voltage across it. The first node listed (or pin one in Capture) is defined as positive. The voltage across the component is therefore defined as the first node voltage, less the second node voltage.

[model name]  
If [model name] is left out, then <value> is the capacitance in farads. If [model name] is specified, then the value is given by the model parameters; see Capacitor value formula.

<initial value>  
The initial voltage across the capacitor during the bias point calculation. It can also be specified in a circuit file using a .IC command as follows:

\[ .\text{IC} \ V(\text{+node, -node}) \ <\text{initial value}> \]

Comments  
Positive current flows from the (+) node through the capacitor to the (-) node. Current flow from the first node through the component to the second node is considered positive.

For details on using the .IC command in a circuit file, see .IC (initial bias point condition) and refer to your PSpice user’s guide for more information.

The initial voltage across the capacitor can also be set in Capture by using the IC1 part if the capacitor is connected to ground or by using the IC2 part for setting the initial conditions between two nodes. These parts can be found in SPECIAL.OLB.

For more information about setting initial conditions, refer to the Capture User’s Guide if you are using Capture, or refer to your PSpice user’s guide if you are using PSpice.
Capture parts

For standard C parts, the effective value of the part is set directly by the VALUE property. For the variable capacitor, C_VAR, the effective value is the product of the base value (VALUE) and multiplier (SET).

In general, capacitors should have positive component values (VALUE property). In all cases, components must not be given a value of zero.

However, there are cases when negative component values are desired. This occurs most often in filter designs that analyze an RLC circuit equivalent to a real circuit. When transforming from the real to the RLC equivalent, it is possible to end up with negative component values.

PSpice A/D allows negative component values for bias point, DC sweep, AC, and noise analyses. A transient analysis may fail for a circuit with negative components. Negative capacitors may create instabilities in time that the analysis cannot handle.

<table>
<thead>
<tr>
<th>Part name</th>
<th>Model type</th>
<th>Property</th>
<th>Property description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>capacitor</td>
<td>VALUE</td>
<td>capacitance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IC</td>
<td>initial voltage across the capacitor during bias point calculation</td>
</tr>
<tr>
<td>C_VAR</td>
<td>VALUE</td>
<td>SET</td>
<td>multiplier</td>
</tr>
</tbody>
</table>

Breakout parts

For non-stock passive and semiconductor devices, Capture provides a set of breakout parts designed for customizing model parameters for simulation. These are useful for setting up Monte Carlo and worst-case analyses with device and/or lot tolerances specified for individual model parameters. Another approach is to use the model editor to derive an instance model and customize this. For example, you could add device and/or lot tolerances to model parameters.

Basic breakout part names consist of the intrinsic PSpice A/D device letter plus the suffix BREAK. By default, the model name is the same as the part name and references the appropriate device model with all parameters set at their default. For instance, the DBREAK part references the DBREAK model which is derived from the intrinsic PSpice A/D D model (.MODEL DBREAK D).

For breakout part CBREAK, the effective value is computed from a formula that is a function of the specified VALUE property.

<table>
<thead>
<tr>
<th>Device type</th>
<th>Part name</th>
<th>Part library</th>
<th>Property</th>
<th>Property description</th>
</tr>
</thead>
<tbody>
<tr>
<td>capacitor</td>
<td>CBREAK</td>
<td>BREAKOUT.OLB</td>
<td>VALUE</td>
<td>capacitance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IC</td>
<td>initial voltage across the capacitor during bias point calculation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MODEL</td>
<td>CAP model name</td>
</tr>
</tbody>
</table>
Capacitor model parameters

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>capacitance multiplier</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>TC1</td>
<td>linear temperature coefficient</td>
<td>°C⁻¹</td>
<td>0.0</td>
</tr>
<tr>
<td>TC2</td>
<td>quadratic temperature coefficient</td>
<td>°C⁻²</td>
<td>0.0</td>
</tr>
<tr>
<td>T_ABS</td>
<td>absolute temperature</td>
<td>°C</td>
<td>0.0</td>
</tr>
<tr>
<td>T_MEASURED</td>
<td>measured temperature</td>
<td>°C</td>
<td>0.0</td>
</tr>
<tr>
<td>T_REL_GLOBAL</td>
<td>relative to current temperature</td>
<td>°C</td>
<td>0.0</td>
</tr>
<tr>
<td>T_REL_LOCAL</td>
<td>relative to AKO model temperature</td>
<td>°C</td>
<td>0.0</td>
</tr>
<tr>
<td>VC1</td>
<td>linear voltage coefficient</td>
<td>volt⁻¹</td>
<td>0.0</td>
</tr>
<tr>
<td>VC2</td>
<td>quadratic voltage coefficient</td>
<td>volt⁻²</td>
<td>0.0</td>
</tr>
</tbody>
</table>

* For information on T_MEASURED, T_ABS, T_REL_GLOBAL, and T_REL_LOCAL, see MODEL (model definition).

Capacitor equations

Capacitor value formula

If [model name] is specified, then the value is given by:

\[
<\text{value}> \cdot C \cdot (1+VC1 \cdot V+VC2 \cdot V^2) \cdot (1+TC1 \cdot (T-T_{\text{nom}})+TC2 \cdot (T-T_{\text{nom}})^2)
\]

where <value> is normally positive (though it can be negative, but not zero). T_{\text{nom}} is the nominal temperature (set using TNOM option).

Capacitor equation for noise

The capacitor does not have a noise model.
Diode

General form

D<name> <(+ node> <(-) node> <model name> [area value]

Examples

DCLAMP 14 0 DMOD
D13 15 17 SWITCH 1.5

Model form

.Model <model name> D [model parameters]

Description

The diode is modeled as an ohmic resistance ($RS$/area) in series with an intrinsic diode. Positive current is current flowing from the anode through the diode to the cathode.

Arguments and options

<(+ node>
The anode.

<(-) node>
The cathode.

[area value]
Scales IS, ISR, IKF, RS, CJ0, and IBV, and has a default value of 1.
IBV and BV are both specified as positive values.
Capture parts

The following table lists the set of diode breakout parts designed for customizing model parameters for simulation. These are useful for setting up Monte Carlo and worst-case analyses with device and/or lot tolerances specified for individual model parameters.

<table>
<thead>
<tr>
<th>Part name</th>
<th>Model type</th>
<th>Property</th>
<th>Property description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBREAK</td>
<td>D, X</td>
<td>MODEL</td>
<td>D model name</td>
</tr>
<tr>
<td>DBREAK3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBREAKCR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBREAKVV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBREAKZ</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Setting operating temperature

Operating temperature can be set to be different from the global circuit temperature by defining one of the model parameters: T_ABS, T_REL_GLOBAL, or T_REL_LOCAL. Additionally, model parameters can be assigned unique measurement temperatures using the T_MEASURED model parameter. For more information, see Special considerations.
# Diode model parameters

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF</td>
<td>flicker noise exponent</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>BV</td>
<td>reverse breakdown knee voltage</td>
<td>volt</td>
<td>infinite</td>
</tr>
<tr>
<td>CJO</td>
<td>zero-bias p-n capacitance</td>
<td>farad</td>
<td>0.0</td>
</tr>
<tr>
<td>EG</td>
<td>bandgap voltage (barrier height)</td>
<td>eV</td>
<td>1.11</td>
</tr>
<tr>
<td>FC</td>
<td>forward-bias depletion capacitance coefficient</td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>IBVL</td>
<td>low-level reverse breakdown knee current</td>
<td>amp</td>
<td>0.0</td>
</tr>
<tr>
<td>IBV</td>
<td>reverse breakdown knee current</td>
<td>amp</td>
<td>1E-10</td>
</tr>
<tr>
<td>IKF</td>
<td>high-injection knee current</td>
<td>amp</td>
<td>infinite</td>
</tr>
<tr>
<td>IS</td>
<td>saturation current</td>
<td>amp</td>
<td>1E-14</td>
</tr>
<tr>
<td>ISR</td>
<td>recombination current parameter</td>
<td>amp</td>
<td>0.0</td>
</tr>
<tr>
<td>KF</td>
<td>flicker noise coefficient</td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td>M</td>
<td>p-n grading coefficient</td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>N</td>
<td>emission coefficient</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>NBV</td>
<td>reverse breakdown ideality factor</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>NBVL</td>
<td>low-level reverse breakdown ideality factor</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>NR</td>
<td>emission coefficient for isr</td>
<td></td>
<td>2.0</td>
</tr>
<tr>
<td>RS</td>
<td>parasitic resistance</td>
<td>ohm</td>
<td>0.0</td>
</tr>
<tr>
<td>TBV1</td>
<td>bv temperature coefficient (linear)</td>
<td>°C&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0.0</td>
</tr>
<tr>
<td>TBV2</td>
<td>bv temperature coefficient (quadratic)</td>
<td>°C&lt;sup&gt;2&lt;/sup&gt;</td>
<td>0.0</td>
</tr>
<tr>
<td>TIKF</td>
<td>ikf temperature coefficient (linear)</td>
<td>°C&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0.0</td>
</tr>
<tr>
<td>TRS1</td>
<td>rs temperature coefficient (linear)</td>
<td>°C&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0.0</td>
</tr>
<tr>
<td>TRS2</td>
<td>rs temperature coefficient (quadratic)</td>
<td>°C&lt;sup&gt;2&lt;/sup&gt;</td>
<td>0.0</td>
</tr>
<tr>
<td>TT</td>
<td>transit time</td>
<td>sec</td>
<td>0.0</td>
</tr>
<tr>
<td>T_ABS</td>
<td>absolute temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_MEASURED</td>
<td>measured temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_REL_GLOBAL</td>
<td>relative to current temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_REL_LOCAL</td>
<td>Relative to AKO model temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>VJ</td>
<td>p-n potential</td>
<td>volt</td>
<td>1.0</td>
</tr>
<tr>
<td>XTI</td>
<td>IS temperature exponent</td>
<td></td>
<td>3.0</td>
</tr>
</tbody>
</table>

* For more information on T_MEASURED, T_ABS, T_REL_GLOBAL, and T_REL_LOCAL, see [MODEL (model definition)](#).
Diode equations

The equations in this section use the following variables:

\[ V_d = \text{voltage across the intrinsic diode only} \]
\[ V_t = k \cdot T/q \text{ (thermal voltage)} \]
\[ k = \text{Boltzmann’s constant} \]
\[ q = \text{electron charge} \]
\[ T = \text{analysis temperature (°K)} \]
\[ T_{\text{nom}} = \text{nominal temperature (set using TNOM option)} \]

Other variables are listed in Diode model parameters.

Diode equations for DC current

\[ I_d = area \cdot (I_{\text{fwd}} - I_{\text{rev}}) \]

\[ I_{\text{fwd}} = \text{forward current} = I_{\text{norm}} \cdot K_{\text{inj}} + I_{\text{rec}} \cdot K_{\text{gen}} \]
\[ I_{\text{norm}} = \text{normal current} = I_S \cdot \left( e^{V_d/(N \cdot V_t)} - 1 \right) \]

if: \[ IKF > 0 \]
then: \[ K_{\text{inj}} = \text{high-injection factor} = \left( \frac{IKF}{IKF + I_{\text{norm}}} \right)^{1/2} \]
else: \[ K_{\text{inj}} = 1 \]

\[ I_{\text{rec}} = \text{recombination current} = I_S R \cdot \left( e^{V_d/(N_R \cdot V_t)} - 1 \right) \]
\[ K_{\text{gen}} = \text{generation factor} = \left( (1 - V_d/V_J)^2 + 0.005 \right)^{3/2} \]

\[ I_{\text{rev}} = \text{reverse current} = I_{\text{rev high}} + I_{\text{rev low}} \]
\[ I_{\text{rev high}} = I_{B_V} \cdot e^{-\left( V_d + B_V \right)/(N_B \cdot V_t)} \]
\[ I_{\text{rev low}} = I_{B_{VL}} \cdot e^{-\left( V_d + B_V \right)/(N_B \cdot V_{VL})} \]

Diode equations for capacitance

\[ C_d = C_t + area \cdot C_j \]

\[ C_t = \text{transit time capacitance} = TT \cdot G_d \]
\[ G_d = \text{DC conductance} = area \cdot \frac{d(I_{\text{norm}} \cdot K_{\text{inj}} + I_{\text{rec}} \cdot K_{\text{gen}})}{dV_d} \]
\[ K_{\text{inj}} = \text{high-injection factor} \]

\[ C_j = C_{JO} \cdot (1 - V_d/V_J)^M \text{ IF: } V_d \leq FC \cdot V_J \]
\[ C_j = C_{JO} \cdot (1 - FC)^{(1+M)} \cdot (1 - FC) \cdot (1+M) \cdot V_d/V_J \text{ IF: } V_d > FC \cdot V_J \]
\[ C_j = \text{junction capacitance} \]
Diode equations for temperature effects

\[ IS(T) = IS \cdot e^{(T/T_{nom}-1) \cdot \frac{Eg(VD)}{N \cdot Vt}} \cdot \left( \frac{T}{T_{nom}} \right)^{XTI/N} \]

\[ ISR(T) = ISR \cdot e^{(T/T_{nom}-1) \cdot \frac{Eg(VD)}{NR \cdot Vt}} \cdot \left( \frac{T}{T_{nom}} \right)^{XTI/NR} \]

\[ IKF(T) = IKF \cdot \left( 1 + \frac{1}{T_{TKF}} \cdot (T - T_{nom}) \right) \]

\[ BV(T) = BV \cdot (1 + TBV1 \cdot (T - T_{nom}) + TBV2 \cdot (T - T_{nom})^2) \]

\[ RS(T) = RS \cdot (1 + TRS1 \cdot (T - T_{nom}) + TRS2 \cdot (T - T_{nom})^2) \]

\[ VJ(T) = VJ \cdot T/T_{nom} - 3 \cdot Vt \cdot ln(T/T_{nom}) - \frac{Eg(T_{nom}) \cdot T}{T_{nom}} + Eg(T) \]

\[ Eg(T) = \text{silicon bandgap energy} = 1.16 - 0.000702 \cdot T^2 / (T + 1108) \]

\[ CJO(T) = CJO \cdot (1 + M \cdot (0.0004 \cdot (T - T_{nom}) + (1 - VJ(T)/VJ))) \]

Diode equations for noise

Noise is calculated assuming a 1.0-hertz bandwidth, using the following spectral power densities (per unit bandwidth).

<table>
<thead>
<tr>
<th>parasitic resistance thermal noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \text{In}^2 = 4 \cdot k \cdot T / (RS \cdot \text{area}) ]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>intrinsic diode shot and flicker noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \text{In}^2 = 2 \cdot q \cdot I_d + KF \cdot I_d \cdot v / \text{FREQUENCY} ]</td>
</tr>
</tbody>
</table>

References

For a detailed description of p-n junction physics, refer to:


Also, for a generally detailed discussion of the U.C. Berkeley SPICE models, including the diode device, refer to:

Voltage-controlled voltage source

Voltage-controlled current source

**General form**

\[ E\text{name} \begin{cases} (+) \text{ node} & \begin{cases} (-) \text{ node} & \begin{cases} (+) \text{ controlling node} \begin{cases} (-) \text{ controlling node} \begin{cases} \text{gain} \\ \text{POLY(<value>)} \\ + \begin{cases} (+) \text{ controlling node} & \begin{cases} (-) \text{ controlling node} \begin{cases} \text{value} \end{cases} \\ \text{coefficient value} \end{cases} \\ \text{transform} \end{cases} \end{cases} \end{cases} \end{cases} \\ \text{VALUE} = \{ \text{expression} \} \end{cases} \\ \text{TABLE} \{ \text{expression} \} = \begin{cases} \text{input value}, & \text{output value} \end{cases} \end{cases} \\ \text{FREQ} \{ \text{expression} \} = \begin{cases} \text{frequency value}, & \text{magnitude value}, & \text{phase value} \end{cases} \end{cases} \end{cases} \end{cases} \]

**Examples**

EBUFF 1 2 10 11 1.0
EAMP 13 0 POLY(1) 26 0 0 500
ENONLIN 100 101 POLY(2) 3 0 4 0.0 13.6 0.2 0.005
ESQREROOT 5 0 VALUE = {5V*SQRT(V(3,2))}
ET2 2 0 TABLE \{V(ANODE,CATHODE)\} = \{(0,0) (30,1)\}
ERC 5 0 LAPLACE \{V(10)\} = \{1/(1+.001*s)\}
ELOWPASS 5 0 FREQ \{V(10)\}=(0,0,0)(5kHz, 0,0)(6kHz -60, 0) DELAY=3.2ms
ELOWPASS 5 0 CHEBSHEV \{V(10)\} = LP 800 1.2K .1dB 50dB
GBUFF 1 2 10 11 1.0
GAMP 13 0 POLY(1) 26 0 0 500
GNONLIN 100 101 POLY(2) 3 0 4 0.0 13.6 0.2 0.005
GPSK 11 6 VALUE = \{5MA*SIN(6.28*10kHz*TIME+V(3))\}
GT ANODE CATHODE VALUE = \{200E-6*PWR(V(1)*V(2),1.5)\}
GLOSSY 5 0 LAPLACE \{V(10)\} = \{exp(-sqrt(C*s*(R+L*s)))\}

**Description**

The voltage-controlled voltage source (E) and the voltage-controlled current source (G) devices have the same syntax. For a voltage-controlled current source just substitute G for E. G generates a current, whereas E generates a voltage.
Arguments and options

POLY(<value>)

Specifies the number of dimensions of the polynomial. The number of pairs of controlling nodes must be equal to the number of dimensions.

(+) and (-) nodes

Output nodes. Positive current flows from the (+) node through the source to the (-) node.

The <(+) controlling node> and <(-) controlling node>

Are in pairs and define a set of controlling voltages. A particular node can appear more than once, and the output and controlling nodes need not be different. The TABLE form has a maximum size of 2048 input/output value pairs.

FREQ

If a DELAY value is specified, the simulator modifies the phases in the FREQ table to incorporate the specified delay value. This is useful for cases of tables which the simulator identifies as being non-causal. When this occurs, the simulator provides a delay value necessary to make the table causal. The new syntax allows this value to be specified in subsequent simulation runs, without requiring the user to modify the table.

If a KEYWORD is specified for FREQ tables, it alters the values in the table. The KEYWORD can be one of the following:

• MAG causes magnitude of frequency response to be interpreted as a raw value instead of dB.
• DB causes magnitude to be interpreted as dB (the default).
• RAD causes phase to be interpreted in radians.
• DEG causes phase to be interpreted in degrees (the default).
• R_I causes magnitude and phase values to be interpreted as real and imaginary magnitudes.

Comments

The first form and the first two examples apply to the linear case; the second form and the third example are for the nonlinear case. The last five forms and examples are analog behavioral modeling (ABM) that have expression, look up table, Laplace transform, frequency response, and filtering. Refer to your PSpice user’s guide for more information on analog behavioral modeling.

Chebyshev filters have two attenuation values, given in dB, which specify the pass band ripple and the stop band attenuation. They can be given in either order, but must appear after all of the cutoff frequencies have been given. Low pass (LP) and high pass (HP) have two cutoff frequencies, specifying the pass band and stop band edges, while band pass (BP) and band reject (BR) filters have four. Again, these can be given in any order.

You can get a list of the filter Laplace coefficients for each stage by enabling the LIST option in the Simulation Settings dialog box. (Click the Options tab, then select the Output file Category and select Device Summary.) The output is written to the .out file after the simulation is complete.

For the linear case, there are two controlling nodes and these are followed by the gain. For all cases, including the nonlinear case (POLY), refer to your PSpice user’s guide.

Expressions cannot be used for linear and polynomial coefficient values in a voltage-controlled voltage source device statement.
Basic SPICE polynomial expressions (POLY)

PSpice A/D (and SPICE) use the following syntax:

```
<controlled source> <connecting nodes> +POLY(<dimension>) <controlling input> <coefficients>
```

where

- `<controlled source>` is `<[E][F][G][H]device name>`, meaning the device type is one of E, F, G, or H
- `<connecting nodes>` specifies `<(+node_name, -node_name)>` pair between which the device is connected
- `<dimension>` is the dimension `<value>` of the polynomial describing the controlling function
- `<controlling input>` specifies `<(+node_name, -node_name)>*` pairs used as input to the voltage controlled source (device types E and G), or `<V device name>*` for the current controlled source (device types F and H), and where the number of controlling inputs for either case equals `<dimension>`
- `<coefficients>` specifies the coefficient `<values>*` for the polynomial transfer function

If the source is one-dimensional (there is only one controlling source), POLY(1) is required unless the linear form is used. If the source is multidimensional (there is more than one controlling source), the dimension needs to be included in the keyword, for instance POLY(2).

Caution must be exercised with the POLY form. For instance,

```
EWRONG 1 0 POLY(1) (1,0) .5 1.0
```

tries to set node 1 to .5 volts greater than node 1. In this case, any analyses which you specify will fail to calculate a result. In particular, PSpice A/D cannot calculate the bias point for a circuit containing EWRONG. This also applies to the VALUE form of EWRONG:

```
(EWRONG 1 0 VALUE = {0.5 * V(1)}).
```

Basic controlled source properties

<table>
<thead>
<tr>
<th>Part name</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>GAIN</td>
<td>gain</td>
</tr>
<tr>
<td>F</td>
<td>GAIN</td>
<td>gain</td>
</tr>
<tr>
<td>G</td>
<td></td>
<td>transconductance</td>
</tr>
<tr>
<td>H</td>
<td></td>
<td>transresistance</td>
</tr>
<tr>
<td>EPOLY, FPOLY, GPOLY, HPOLY</td>
<td>COEFF</td>
<td>polynomial coefficient</td>
</tr>
</tbody>
</table>

PSpice A/D has a built-in capability allowing controlled sources to be defined with a polynomial transfer function of any degree and any dimension. Polynomials have associated
coefficients for each term. Consider a voltage-controlled source with voltages $V_1$, $V_2$, ... $V_n$.

The coefficients are associated with the polynomial according to this convention:

\[
V_{\text{out}} = P_0 + P_1 \cdot V_1 + P_2 \cdot V_2 + \cdots + P_n \cdot V_n + \\
\vdots \\
\vdots \\
P_{n!/(2(n-2)!)} + 2n \cdot V_n \cdot V_n + \\
P_{n!/(2(n-2)!)}(2n+1) \cdot V_n \cdot V_n + \\
\vdots \\
\vdots
\]

The above is written for a voltage-controlled voltage source, but the form is similar for the other sources.

The POLY device types shown in Basic controlled source properties are defined with a dimension of one, meaning there is only one controlling source. However, similar devices can be defined of any degree and dimension by creating parts with appropriate coefficient and TEMPLATE properties and the appropriate number of input pins.

The current-controlled device models (F, FPOLY, H, and HPOLY) contain a current-sensing voltage source. When netlisted, they generate two device declarations to the circuit file set: one for the controlled source and one for the independent current-sensing voltage source.

When defining a current-controlled source part of higher dimension, the TEMPLATE property must account for the same number of current-sensing voltage sources (equal to the dimension value). For example, a two dimensional current-controlled voltage source is described by the following polynomial equation:

\[
V_{\text{out}} = C_0 + C_1 I_1 + C_2 I_2 + C_{11} I_1^2 + C_{12} I_1 I_2 + C_{22} I_2^2
\]

To create the two dimensional HPOLY2 part, these properties must be defined:

\begin{verbatim}
COEFF0 = 1
COEFF1 = 1
COEFF2 = 1
COEFF11 = 1
COEFF12 = 1
COEFF22 = 1
COEFFS = @COEFF0 @COEFF1 @COEFF2 @COEFF11 @COEFF12 @COEFF22
TEMPLATE = H^@REFDES %5 %6 POLY(2) VH1^@REFDES VH2^@REFDES
\n+ @COEFFS 
VH1^@REFDES %1 %2 0V 
VH2^@REFDES %3 %4 0V
\end{verbatim}

The TEMPLATE definition is actually contained on a single line. The VH1 and VH2 fragments after the \n characters represent the device declarations for the two current-sensing voltage sources required by this part. Also, the part graphics must have the appropriate number of pins. When placing an instance of HPOLY2 in your schematic, the COEFF$n$ properties must be appropriately set.

### Implementation examples

Following are some examples of traditional SPICE POLY constructs and equivalent ABM parts which could be used instead.
Example 1: four-input voltage adder

This is an example of a device which takes four input voltages and sums them to provide a single output voltage.

The representative polynomial expression would be as follows:

\[ V_{\text{out}} = 0.0 + (1.0)V_1 + (1.0)V_2 + (1.0)V_3 + (1.0)V_4 \]

The corresponding SPICE POLY form would be as follows:

```
ESUM 100 101 POLY(4) (1,0) (2,0) (3,0) (4,0) 0.0 1.0 1.0 + 1.0 1.0
```

This could be represented with a single ABM expression device configured with the following expression properties:

\[
\begin{align*}
\text{EXP1} & = V(1,0) + \\
\text{EXP2} & = V(2,0) + \\
\text{EXP3} & = V(3,0) + \\
\text{EXP4} & = V(4,0)
\end{align*}
\]

Following template substitution for the ABM device, the output becomes:

\[ V(\text{OUT}) = \{ V(1,0) + V(2,0) + V(3,0) + V(4,0) \} \]

Example 2: two-input voltage multiplier

This is an example of a device which takes two input voltages and multiplies them together resulting in a single output voltage.

The representative polynomial expression would be as follows:

\[ V_{\text{out}} = 0.0 + (0.0)V_1 + (0.0)V_2 + (0.0)V_1^2 + (1.0)V_1V_2 \]

The corresponding SPICE POLY form would be as follows:

```
EMULT 100 101 POLY(2) (1,0) (2,0) 0.0 0.0 0.0 0.0 1.0
```

This could be represented with a single MULT device. For additional examples of a voltage multiplier device, refer to the Analog Behavioral Modeling chapter of your PSpice user’s guide.

Example 3: voltage squarer

This is an example of a device that outputs the square of the input value.

For the one-dimensional polynomial, the representative polynomial expression reduces to:

\[ V_{\text{out}} = P_0 + P_1V + P_2V^2 + \ldots + P_nV^n \]

The corresponding SPICE POLY form would be as follows:

```
ESQUARE 100 101 POLY(1) (1,0) 0.0 0.0 1.0
```

This could be represented by a single instance of the MULT part, with both inputs from the same net. This results in the following:

\[ V_{\text{out}} = (V_{\text{in}})^2 \]
Current-controlled current source

Current-controlled voltage source

General form

F<name> <(+> node> <(-) node>
+ <controlling V device name> <gain>
F<name> <(+> node> <(-) node> POLY(<value>)
+ <controlling V device name>*
+ < <polynomial coefficient value> >*

Examples

FSENSE 1 2 VSENSE 10.0
FAMP 13 0 POLY(1) VIN 0 500
FNONLIN 100 101 POLY(2) VCNTRL1 VCINTRL2 0.0 13.6 0.2 0.005

Description

The Current-Controlled Current Source (F) and the Current-Controlled Voltage Source (H) devices have the same syntax. For a Current-Controlled Voltage Source just substitute an H for the F. The H device generates a voltage, whereas the F device generates a current.

Arguments and options

(+ and -)

Output nodes. A positive current flows from the (+) node through the source to the (-) node. The current through the controlling voltage source determines the output current. The controlling source must be an independent voltage source (V device), although it need not have a zero DC value.

POLY(<value>)

Specifies the number of dimensions of the polynomial. The number of controlling voltage sources must be equal to the number of dimensions.

Comments

The first General Form and the first two examples apply to the linear case. The second form and the last example are for the nonlinear case.

For the linear case, there must be one controlling voltage source and its name is followed by the gain. For all cases, including the nonlinear case (POLY), refer to your PSpice user’s guide.

In a current-controlled current source device statement, expressions cannot be used for linear and polynomial coefficient values.

Basic SPICE polynomial expressions (POLY)

For more information on the POLY form, see Basic SPICE polynomial expressions (POLY).
Independent current source & stimulus

Independent voltage source & stimulus

General form

\[ I<\text{name}> <(+))\text{ node}> <(-)\text{ node}>\]
+ [ [DC] <value> ]
+ [ AC <magnitude value> [phase value] ]
+ [STIMULUS=<stimulus name>]
+ [transient specification]

Examples

IBIAS 13 0 2.3mA
IAC 2 3 AC .001
IACPHS 2 3 AC .001 90
IPULSE 1 0 PULSE(-1mA 1mA 2ns 2ns 2ns 50ns 100ns)
I3 26 77 DC .002 AC 1 SIN(.002 .002 1.5MEG)

Description

This element is a current source. Positive current flows from the (+) node through the source to the (-) node: in the first example, IBIAS drives node 13 to have a negative voltage. The default value is zero for the DC, AC, and transient values. None, any, or all of the DC, AC, and transient values can be specified. The AC phase value is in degrees. The pulse and exponential examples are explained later in this section.

The independent current source & stimulus (I) and the independent voltage source & stimulus (V) devices have the same syntax. For an independent voltage source & stimulus just substitute a V for the I. The V device functions identically and has the same syntax as the I device, except that it generates voltage instead of current.

The variables TSTEP and TSTOP, which are used in defaulting some waveform parameters, are set by the .TRAN (transient analysis) command. TSTEP is <print step value> and TSTOP is <final time value>. The .TRAN command can be anywhere in the circuit file; it need not come after the voltage source.
Arguments and options

<stimulus name>
References a [STIMULUS (stimulus)] definition.

[transient specification]

<table>
<thead>
<tr>
<th>Use this value...</th>
<th>To produce this result...</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXP (&lt;parameters&gt;)</td>
<td>an exponential waveform</td>
</tr>
<tr>
<td>PULSE (&lt;parameters&gt;)</td>
<td>a pulse waveform</td>
</tr>
<tr>
<td>PWL (&lt;parameters&gt;)</td>
<td>a piecewise linear waveform</td>
</tr>
<tr>
<td>SFFM (&lt;parameters&gt;)</td>
<td>a frequency-modulated waveform</td>
</tr>
<tr>
<td>SIN (&lt;parameters&gt;)</td>
<td>a sinusoidal waveform</td>
</tr>
</tbody>
</table>
Independent current source & stimulus (EXP)

General form

\[ \text{EXP} \left( \langle i1 \rangle \ <i2\rangle \ <td1\rangle \ <tc1\rangle \ <td2\rangle \ <tc2\rangle \right) \]

Examples

IRAMP 10 5 EXP(1 5 1 .2 2 .5)

Waveform parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>\langle i1 \rangle</td>
<td>Initial current</td>
<td>amp</td>
<td>none</td>
</tr>
<tr>
<td>\langle i2 \rangle</td>
<td>Peak current</td>
<td>amp</td>
<td>none</td>
</tr>
<tr>
<td>\langle td1 \rangle</td>
<td>Rise (fall) delay</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>\langle tc1 \rangle</td>
<td>Rise (fall) time constant</td>
<td>sec</td>
<td>TSTEP</td>
</tr>
<tr>
<td>\langle td2 \rangle</td>
<td>Fall (rise) delay</td>
<td>sec</td>
<td>\langle td1 \rangle + TSTEP</td>
</tr>
<tr>
<td>\langle tc2 \rangle</td>
<td>Fall (rise) time constant</td>
<td>sec</td>
<td>TSTEP</td>
</tr>
</tbody>
</table>

Description

The EXP form causes the current to be \langle i1 \rangle for the first \langle td1 \rangle seconds. Then, the current decays exponentially from \langle i1 \rangle to \langle i2 \rangle using a time constant of \langle tc1 \rangle. The decay lasts \langle td2 \rangle-\langle td1 \rangle seconds. Then, the current decays from \langle i2 \rangle back to \langle i1 \rangle using a time constant of \langle tc2 \rangle. Independent current source and stimulus exponential waveform formulas describe the EXP waveform.

Independent current source and stimulus exponential waveform formulas

<table>
<thead>
<tr>
<th>Time period</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 to \langle td1 \rangle</td>
<td>\langle i1 \rangle</td>
</tr>
<tr>
<td>\langle td1 \rangle to \langle td2 \rangle</td>
<td>\langle i1 \rangle + (\langle i2 \rangle-\langle i1 \rangle) \cdot (1-e^{-\langle TIME-td1 \rangle/\langle tc1 \rangle})</td>
</tr>
<tr>
<td>\langle td2 \rangle to TSTOP</td>
<td>\langle i1 \rangle + (\langle i2 \rangle-\langle i1 \rangle) \cdot ((1-e^{-\langle TIME-td1 \rangle/\langle tc1 \rangle})- (1-e^{-\langle TIME-td2 \rangle/\langle tc2 \rangle}))</td>
</tr>
</tbody>
</table>
Independent current source & stimulus (PULSE)

**General form**

PULSE ((<i1> <i2> <td> <tr> <tf> <pw> <per>))

**Example**

ISW 10 5 PULSE(1A 5A 1sec .1sec .4sec .5sec 2sec)

**Waveform parameters**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;i1&gt;</td>
<td>Initial current</td>
<td>amp</td>
<td>none</td>
</tr>
<tr>
<td>&lt;i2&gt;</td>
<td>Pulsed current</td>
<td>amp</td>
<td>none</td>
</tr>
<tr>
<td>&lt;td&gt;</td>
<td>Delay</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>&lt;tf&gt;</td>
<td>Fall time</td>
<td>sec</td>
<td>TSTEP</td>
</tr>
<tr>
<td>&lt;tr&gt;</td>
<td>Rise time</td>
<td>sec</td>
<td>TSTEP</td>
</tr>
<tr>
<td>&lt;pw&gt;</td>
<td>Pulse width</td>
<td>sec</td>
<td>TSTOP</td>
</tr>
<tr>
<td>&lt;per&gt;</td>
<td>Period</td>
<td>sec</td>
<td>TSTOP</td>
</tr>
</tbody>
</table>

**Description**

The PULSE form causes the current to start at <i1>, and stay there for <td> seconds. Then, the current goes linearly from <i1> to <i2> during the next <tr> seconds, and then the current stays at <i2> for <pw> seconds. Then, it goes linearly from <i2> back to <i1> during the next <tf> seconds. It stays at <i1> for per-(tr+pw+tf) seconds, and then the cycle is repeated except for the initial delay of <td> seconds. [Independent current source and stimulus pulse waveform formulas](#) describe the PULSE waveform.
### Independent current source and stimulus pulse waveform formulas

<table>
<thead>
<tr>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(i_1)</td>
</tr>
<tr>
<td>(t_d)</td>
<td>(i_1)</td>
</tr>
<tr>
<td>(t_d+tr)</td>
<td>(i_2)</td>
</tr>
<tr>
<td>(t_d+tr+pw)</td>
<td>(i_2)</td>
</tr>
<tr>
<td>(t_d+tr+pw+tf)</td>
<td>(i_1)</td>
</tr>
<tr>
<td>(t_d+per)</td>
<td>(i_1)</td>
</tr>
<tr>
<td>(t_d+per+tr)</td>
<td>(i_2)</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>
Independent current source & stimulus (PWL)

**General form**

PWL
+ [TIME_SCALE_FACTOR=<value>]
+ [VALUE_SCALE_FACTOR=<value>]
+ (corner_points)*

where corner_points are:

\(<\text{tn}, \text{in}>\)  to specify a point
FILE <filename> to read point values from a file
REPEAT FOR <n> (corner_points)*  to repeat <n> times
REPEAT FOREVER (corner_points)*  to repeat forever

**Examples**

v1 1 2 PWL (0,1) (1.2,5) (1.4,2) (2,4) (3,1)

v2 3 4 PWL REPEAT FOR 5 (1,0) (2,1) (3,0) ENDREPEAT
+ ENDREPEAT
v3 5,6 PWL REPEAT FOR 5 FILE DATA1.TAB
+ ENDREPEAT
v4 7 8 PWL TIME_SCALE_FACTOR=0.1
+ REPEAT FOREVER
+ REPEAT FOR 5 (1,0) (2,1) (3,0) ENDREPEAT
+ ENDREPEAT
+ ENDREPEAT

n volt square wave (where n is 1, 2, 3, 4, then 5); 75% duty cycle; 10 cycles; 1 microseconds per cycle:

.PARAM N=1
.STEP PARAM N 1,5,1
V1 1 0 PWL
+ TIME_SCALE_FACTOR=1e-6 ; all time units are scaled to microseconds
+ REPEAT FOR 10
+ (.25, 0)(.26, {N})(.99, {N})(1, 0)
+ ENDREPEAT

5 volt square wave; 75% duty cycle; 10 cycles; 10 microseconds per cycle; followed by 50% duty cycle n volt square wave (where n is 1, 2, 3, 4, then 5) lasting until the end of simulation:

.PARAM N=.2
.STEP PARAM N .2, 1.0, .2
V1 1 0 PWL
+ TIME_SCALE_FACTOR=1e-5 ; all time units are scaled to 10 us
+ VALUE_SCALE_FACTOR=5
+ REPEAT FOR 10
+ (.25, 0)(.26, 1)(.99, 1)(1, 0)
+ ENDREPEAT

+ REPEAT FOREVER
+ (+.50, 0)
+ (+.01, {N}) ; iteration time .51
+ (+.48, {N}) ; iteration time .99
+ (1, 0)
+ ENDREPEAT
Assuming that a PWL specification has been given for a device to generate two triangular waveforms:

```
V3 1 0 PWL (1ms, 1)(2ms, 0)(3ms, 1)(4ms, 0)
```

Or, to replace the above with

```
V3 1 0 PWL FILE TRIANGLE.IN
```

where the file `triangle.in` would need to contain:

```
(1ms, 1)(2ms, 0)(3ms, 1)(4ms, 0)
```

### Waveform parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;tn&gt;</td>
<td>time at corner</td>
<td>seconds</td>
<td>none</td>
</tr>
<tr>
<td>&lt;vn&gt;</td>
<td>voltage at corner</td>
<td>volts</td>
<td>none</td>
</tr>
<tr>
<td>&lt;n&gt;</td>
<td>number of repetitions</td>
<td>positive integer, 0, or -1</td>
<td>none</td>
</tr>
</tbody>
</table>

* <tn> and <n> cannot be expressions; <vn> may be an expression.

### Description

The PWL form describes a piecewise linear waveform. Each pair of time-current values specifies a corner of the waveform. The current at times between corners is the linear interpolation of the currents at the corners.

![Waveform diagram]

### Arguments and options

- `<time_scale_factor>` and/or `<value_scale_factor>`
  
  Can be included immediately after the PWL keyword to show that the time and/or current value pairs are to be multiplied by the appropriate scale factor. These scale factors can be expressions, in which case they are evaluated once per outer simulation loop, and thus should be composed of expressions not containing references to voltages or currents.

- `<tn>` and `<in>`
  
  The transient specification corner points for the PWL waveform, as shown in the first example. The `<in>` can be an expression having the same restrictions as the scaling keywords, but `<tn>` must be a literal.
<file name>
The text file that supplies the time-current (<tn> <in>) pairs. The contents of this file are read by the same parser that reads the circuit file, so that engineering units (e.g., 10us) are correctly interpreted. Note that the continuation + signs in the first column are unnecessary and therefore discouraged.

A typical file can be created by editing an existing PWL specification, replacing all + signs with blanks (to avoid unintentional +time). Only numbers (with units attached) can appear in the file; expressions for <tn> and <n> values are invalid. All absolute time points in <file name> are with respect to the last (<tn> <in>) entered. All relative time points are with respect to the last time point.

REPEAT ... ENDREPEAT
These loops permit repetitions.

They can appear anywhere a (<tn> <in>) pair can appear. Absolute times within REPEAT loops are with respect to the start of the current iteration. The REPEAT ... ENDREPEAT specifications can be nested to any depth. Make sure that the current value associated with the beginning and ending time points (within the same REPEAT loop or between adjacent REPEAT loops), are the same when 0 is specified as the first point in a REPEAT loop.

<n>
A REPEAT FOR -1 ... ENDREPEAT is treated as if it had been REPEAT FOREVER ... ENDREPEAT. A REPEAT FOR 0 ... ENDREPEAT is ignored (other than syntax checking of the enclosed corner points).
Independent current source & stimulus (SFFM)

General form

SFFM (<ioff> <iampl> <fc> <mod> <fm>)

Example

IMOD 10 5 SFFM(2 1 8Hz 4 1Hz)

Waveform parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;ioff&gt;</td>
<td>offset current</td>
<td>amp</td>
<td>none</td>
</tr>
<tr>
<td>&lt;iampl&gt;</td>
<td>peak amplitude of current</td>
<td>amp</td>
<td>none</td>
</tr>
<tr>
<td>&lt;fc&gt;</td>
<td>carrier frequency</td>
<td>hertz</td>
<td>1/TSTOP</td>
</tr>
<tr>
<td>&lt;mod&gt;</td>
<td>modulation index</td>
<td>hertz</td>
<td>0</td>
</tr>
<tr>
<td>&lt;fm&gt;</td>
<td>modulation frequency</td>
<td>hertz</td>
<td>1/TSTOP</td>
</tr>
</tbody>
</table>

Description

The SFFM (Single-Frequency FM) form causes the current, as illustrated below, to follow the formula:

\[ i_{off} + i_{ampl} \cdot \sin(2\pi \cdot fc \cdot TIME + mod \cdot \sin(2\pi \cdot fm \cdot TIME) ) \]
Independent current source & stimulus (SIN)

General form  
\[ \text{SIN (<ioff> <iampl> <freq> <td> <df> <phase>)} \]

Examples  
\[ \text{ISIG 10 5 SIN(2 2 5Hz 1sec 1 30)} \]

Waveform parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;ioff&gt;</td>
<td>offset current</td>
<td>amp</td>
<td>none</td>
</tr>
<tr>
<td>&lt;iampl&gt;</td>
<td>peak amplitude of current</td>
<td>amp</td>
<td>none</td>
</tr>
<tr>
<td>&lt;freq&gt;</td>
<td>frequency</td>
<td>hertz</td>
<td>1/TSTOP</td>
</tr>
<tr>
<td>&lt;td&gt;</td>
<td>delay</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>&lt;df&gt;</td>
<td>damping factor</td>
<td>sec(^{-1})</td>
<td>0</td>
</tr>
<tr>
<td>&lt;phase&gt;</td>
<td>phase</td>
<td>degree</td>
<td>0</td>
</tr>
</tbody>
</table>

Description  
The sinusoidal (SIN) waveform causes the current to start at <ioff> and stay there for <td> seconds.

Then, the current becomes an exponentially damped sine wave. Independent current source and stimulus sinusoidal waveform formulas describe the SIN waveform.

The SIN waveform is for transient analysis only. It does not have any effect on AC analysis. To give a value to a current during AC analysis, use an AC specification, such as:

\[ \text{IAC 3 0 AC 1mA} \]

where IAC has an amplitude of one milliampere during AC analysis, and can be zero during transient analysis. For transient analysis use, for example:

\[ \text{ITRAN 3 0 SIN(0 1mA 1kHz)} \]

where ITRAN has an amplitude of one milliampere during transient analysis and is zero during AC analysis. Refer to your PSpice user’s guide.
### Independent current source and stimulus sinusoidal waveform formulas

<table>
<thead>
<tr>
<th>Time period</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>to TSTOP</td>
<td>(i_{off} \cdot i_{amp} \cdot \sin(2\pi \cdot \frac{(freq \cdot (TIME-t_d)+phase)}{360^\circ}) \cdot e^{-(TIME-t_d) \cdot df} )</td>
</tr>
<tr>
<td>to td</td>
<td>(i_{off} \cdot i_{amp} \cdot \sin(2\pi \cdot \frac{phase}{360^\circ}) )</td>
</tr>
</tbody>
</table>
**Junction FET**

**General form**

```
J<name> <drain node> <gate node> <source node> <model name> +[area value]
```

**Examples**

```
JIN 100 1 0 JFAST
J13 22 14 23 JNOM 2.0
```

**Model form**

```
.MODEL <model name> NJF [model parameters]
.MODEL <model name> PJF [model parameters]
```

**Description**

The JFET is modeled as an intrinsic FET using an ohmic resistance (RD/area) in series with the drain, and using another ohmic resistance (RS/area) in series with the source. Positive current is current flowing into a terminal.

**Arguments and options**

```
[area value]
```

The relative device area. It has a default value of 1.0.
Capture parts

The following table lists the set of JFET breakout parts designed for customizing model parameters for simulation. These are useful for setting up Monte Carlo and worst-case analyses with device and/or lot tolerances specified for individual model parameters.

<table>
<thead>
<tr>
<th>Part name</th>
<th>Model type</th>
<th>Property</th>
<th>Property description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBREAKN</td>
<td>NJF</td>
<td>AREA</td>
<td>area scaling factor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MODEL</td>
<td>NJF model name</td>
</tr>
<tr>
<td>JBREAKP</td>
<td>PJF</td>
<td>AREA</td>
<td>area scaling factor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MODEL</td>
<td>PJF model name</td>
</tr>
</tbody>
</table>

Setting operating temperature

Operating temperature can be set to be different from the global circuit temperature by defining one of the model parameters: T_ABS, T_REL_GLOBAL, or T_REL_LOCAL. Additionally, model parameters can be assigned unique measurement temperatures using the T_MEASURED model parameter. For more information, see Model parameters.
## Model parameters

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF</td>
<td>flicker noise exponent</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ALPHA</td>
<td>ionization coefficient</td>
<td>volt⁻¹</td>
<td>0</td>
</tr>
<tr>
<td>BETA</td>
<td>transconductance coefficient</td>
<td>amp/volt⁻²</td>
<td>1E-4</td>
</tr>
<tr>
<td>BETATCE</td>
<td>BETA exponential temperature coefficient</td>
<td>%/°C</td>
<td>0</td>
</tr>
<tr>
<td>CGD</td>
<td>zero-bias gate-drain p-n capacitance</td>
<td>farad</td>
<td>0</td>
</tr>
<tr>
<td>CGS</td>
<td>zero-bias gate-source p-n capacitance</td>
<td>farad</td>
<td>0</td>
</tr>
<tr>
<td>FC</td>
<td>forward-bias depletion capacitance coefficient</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>IS</td>
<td>gate p-n saturation current</td>
<td>amp</td>
<td>1E-14</td>
</tr>
<tr>
<td>ISR</td>
<td>gate p-n recombination current parameter</td>
<td>amp</td>
<td>0</td>
</tr>
<tr>
<td>KF</td>
<td>flicker noise coefficient</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>LAMBDA</td>
<td>channel-length modulation</td>
<td>volt⁻¹</td>
<td>0</td>
</tr>
<tr>
<td>M</td>
<td>gate p-n grading coefficient</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>gate p-n emission coefficient</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>NR</td>
<td>emission coefficient for isr</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>PB</td>
<td>gate p-n potential</td>
<td>volt</td>
<td>1.0</td>
</tr>
<tr>
<td>RD</td>
<td>drain ohmic resistance</td>
<td>ohm</td>
<td>0</td>
</tr>
<tr>
<td>RS</td>
<td>source ohmic resistance</td>
<td>ohm</td>
<td>0</td>
</tr>
<tr>
<td>T_ABS</td>
<td>absolute temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_MEASURED</td>
<td>measured temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_REL_GLOBAL</td>
<td>relative to current temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_REL_LOCAL</td>
<td>relative to AKO model temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>VK</td>
<td>ionization knee voltage</td>
<td>volt</td>
<td>0</td>
</tr>
<tr>
<td>VTO</td>
<td>threshold voltage</td>
<td>volt</td>
<td>-2.0</td>
</tr>
<tr>
<td>VTOTC</td>
<td>VTO temperature coefficient</td>
<td>volt/°C</td>
<td>0</td>
</tr>
<tr>
<td>XTI</td>
<td>IS temperature coefficient</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>


**VTO < 0** means the device is a depletion-mode JFET (for both N-channel and P-channel) and **VTO > 0** means the device is an enhancement-mode JFET. This conforms to U.C. Berkeley SPICE.
**JFET equations**

The equations in this section describe an N-channel JFET. For P-channel devices, reverse the sign of all voltages and currents.

The following variables are used:

- $V_{gs} = \text{intrinsic gate-intrinsic source voltage}$
- $V_{gd} = \text{intrinsic gate-intrinsic drain voltage}$
- $V_{ds} = \text{intrinsic drain-intrinsic source voltage}$
- $C_{gs} = \text{gate-source capacitance}$
- $C_{gd} = \text{gate-drain capacitance}$
- $V_t = k \cdot T/q$ (thermal voltage)
- $k = \text{Boltzmann’s constant}$
- $q = \text{electron charge}$
- $T = \text{analysis temperature (°K)}$
- $T_{nom} = \text{nominal temperature (set using TNOM option)}$

Other variables are listed in [Model parameters](#).

Positive current is current flowing into a terminal (for example, positive drain current flows from the drain through the channel to the source).
JFET equations for DC current

<table>
<thead>
<tr>
<th>all levels</th>
</tr>
</thead>
</table>

\[
\text{Ig} = \text{gate current} = \textit{area} \cdot (\text{Igs} + \text{Igd})
\]

\[
\text{Igs} = \text{gate-source leakage current} = \text{In} + \text{Ir} \cdot \text{Kg}
\]

\[
\text{In} = \text{normal current} = \textit{IS} \cdot (e^{\frac{\text{Vgs}}{\text{N} \cdot \text{Vt}}} - 1)
\]

\[
\text{Ir} = \text{recombination current} = \textit{ISR} \cdot (e^{\frac{\text{Vgs}}{\text{NR} \cdot \text{Vt}}} - 1)
\]

\[
\text{Kg} = \text{generation factor} = ((1 - \frac{\text{Vgs}}{\text{PB}})^2 + 0.005)^{\frac{M}{2}}
\]

\[
\text{Igd} = \text{gate-drain leakage current} = \text{In} + \text{Ir} \cdot \text{Kg} + \text{Ii}
\]

\[
\text{In} = \text{normal current} = \textit{IS} \cdot (e^{\frac{\text{Vgd}}{\text{N} \cdot \text{Vt}}} - 1)
\]

\[
\text{Ir} = \text{recombination current} = \textit{ISR} \cdot (e^{\frac{\text{Vgd}}{\text{NR} \cdot \text{Vt}}} - 1)
\]

\[
\text{Kg} = \text{generation factor} = ((1 - \frac{\text{Vgd}}{\text{PB}})^2 + 0.005)^{\frac{M}{2}}
\]

\[
\text{Ii} = \text{impact ionization current}
\]

\[
\text{for forward saturation region:}
\]

\[
0 < \text{Vgs} - \text{VTO} < \text{Vds}
\]

\[
\text{then:}
\]

\[
\text{Ii} = \text{Idrain} \cdot \text{ALPHA} \cdot \text{vdif} \cdot e^{\frac{-\text{VK}}{\text{vdif}}}
\]

\[
\text{where}
\]

\[
\text{vdif} = \text{Vds} - (\text{Vgs} - \text{VTO})
\]

\[
\text{else:}
\]

\[
\text{Ii} = 0
\]

\[
\text{Id} = \text{drain current} = \textit{area} \cdot (\text{Idrain} - \text{Igd})
\]

\[
\text{Is} = \text{source current} = \textit{area} \cdot (-\text{Idrain} - \text{Igs})
\]
JFET equations for DC current (continued)

<table>
<thead>
<tr>
<th>all levels: Idrain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal mode: Vds ≥ 0</td>
</tr>
<tr>
<td>Case 1</td>
</tr>
<tr>
<td>for cutoff region: Vgs-VTO ≤ 0</td>
</tr>
<tr>
<td>then: Idrain = 0</td>
</tr>
<tr>
<td>Case 2</td>
</tr>
<tr>
<td>for linear region: Vds ≤ Vgs-VTO</td>
</tr>
<tr>
<td>then: Idrain = BETA·(1+LAMBDA·Vds)·Vds·(2·(Vgs-VTO)-Vds)</td>
</tr>
<tr>
<td>Case 3</td>
</tr>
<tr>
<td>for saturation region: 0 &lt; Vgs-VTO &lt; Vds</td>
</tr>
<tr>
<td>then: Idrain = BETA·(1+LAMBDA·Vds)·(Vgs-VTO)^2</td>
</tr>
</tbody>
</table>

Inverted mode: Vds < 0

Switch the source and drain in the normal mode equations above.

JFET equations for capacitance

All capacitances are between terminals of the intrinsic JFET (that is, to the inside of the ohmic drain and source resistances).

<table>
<thead>
<tr>
<th>gate-source depletion capacitance</th>
</tr>
</thead>
<tbody>
<tr>
<td>For: Vgs ≤ FC·PB</td>
</tr>
<tr>
<td>Cgs = area·CGS·(1-Vgs/PB)^M</td>
</tr>
<tr>
<td>For: Vgs &gt; FC·PB</td>
</tr>
<tr>
<td>Cgs = area·CGS·(1-FC)^M·(1-FC·(1+M)+M·Vgs/PB)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>gate-drain depletion capacitance</th>
</tr>
</thead>
<tbody>
<tr>
<td>For: Vgd ≤ FC·PB</td>
</tr>
<tr>
<td>Cgd = area·CGD·(1-Vgd/PB)^M</td>
</tr>
<tr>
<td>For: Vgd &gt; FC·PB</td>
</tr>
<tr>
<td>Cgd = area·CGD·(1-FC)^M·(1-FC·(1+M)+M·Vgd/PB)</td>
</tr>
</tbody>
</table>
JFET equations for temperature effects

The drain and source ohmic (parasitic) resistances have no temperature dependence.

\[
\begin{align*}
V_{TO}(T) &= V_{TO} + V_{TOTC} \cdot (T - T_{nom}) \\
BETA(T) &= BETA \cdot 1.01 \cdot BETA_{TCE} \cdot (T - T_{nom}) \\
I_{S}(T) &= I_{S} \cdot e^{\left(\frac{T - T_{nom}}{1.01} - \frac{BETA_{TCE} \cdot (T - T_{nom})}{XTIN} \right)} \\
I_{SR}(T) &= I_{SR} \cdot e^{\left(\frac{T - T_{nom}}{1.01} - \frac{BETA_{TCE} \cdot (T - T_{nom})}{XTIN} \right)} \\
PB(T) &= PB \cdot T/T_{nom} - 3 \cdot V_{t} \cdot ln(T/T_{nom}) - Eg(T_{nom}) \cdot T/T_{nom} + Eg(T) \\
CGS(T) &= CGS \cdot \left(1 + M \cdot (0.0004 \cdot (T - T_{nom}) + (1 - PB(T)/PB))\right) \\
CGD(T) &= CGD \cdot \left(1 + M \cdot (0.0004 \cdot (T - T_{nom}) + (1 - PB(T)/PB))\right)
\end{align*}
\]

where \(Eg(T) = \text{silicon bandgap energy} = 1.16 - 0.000702 \cdot T^2/(T+1108)\)

\[
\begin{align*}
VR(\text{parasitic resistance thermal noise}) & = 4 \cdot k \cdot T/(RS/\text{area}) \\
Id^2 & = 4 \cdot k \cdot T/(RD/\text{area})
\end{align*}
\]

\[
\begin{align*}
I_{\text{Drain}} &= 4 \cdot k \cdot T \cdot \text{gm} \cdot 2/3 + KF \cdot I_{\text{Drain}^{\text{avg}}} \cdot F\text{REQUENCY} \\
& \quad \text{where gm = } dI_{\text{Drain}}/dV_{gs} \text{ (at the DC bias point)}
\end{align*}
\]

Reference

For more information about the U.C. Berkeley SPICE models, including the JFET device, refer to:

Inductor coupling (and magnetic core)

Transmission line coupling

**General form**

\[ K\text{name} \ L\text{inductor name} \ L\text{inductor name}^* \ \text{coupling value} \]

\[ K\text{name} \ L\text{inductor name}^* \ \text{coupling value} \ \text{model name} \ [\text{size value}] \]

\[ K\text{name}\ T\text{transmission line name}\ T\text{transmission line name} \]

+ \text{Cm=capacitive coupling} \ Lm=\text{inductive coupling} \]

**Examples**

KTUNED L3OUT L4IN .8

KTRANSFRM LPRIMARY LSECONDARY 1

KXFRM L1 L2 L3 L4 .98 KPOT_3C8

K2LINES T1 T2 Lm=1m Cm=.5p

**Model form**

.MODEL <model name> CORE [model parameters]

**Description**

This device can be used to define coupling between inductors (transformers) or between transmission lines. This device also refers to a nonlinear magnetic core (CORE) model to include magnetic hysteresis effects in the behavior of a single inductor (winding), or in multiple coupled windings.
Inductor coupling

Arguments and options

K<name> L<inductor name>
Coupled two or more inductors.

Place a period (.) on the first node of each inductor. For example:

\[ \begin{align*}
  & L1 1 0 AC 1mA \\
  & L1 1 0 10uH \\
  & L2 2 0 10uH \\
  & R2 2 0 .1 \\
  & K12 L1 L2 1
\end{align*}\]

The current through \( L2 \) is in the opposite direction as the current through \( L1 \). The polarity is determined by the order of the nodes in the \( L \) devices and not by the order of inductors in the \( K \) statement.

<coupling value>
This is the coefficient of mutual coupling, which must be between -1.0 and 1.0.

This coefficient is defined by the equation

\[ <\text{coupling value}> = \frac{M_{ij}}{(L_i \cdot L_j)^{1/2}} \]

where

\( L_i, L_j \) = a coupled-pair of inductors

\( M_{ij} \) = the mutual inductance between \( L_i \) and \( L_j \)

For transformers of normal geometry, use 1.0 as the value. Values less than 1.0 occur in air core transformers when the coils do not completely overlap.

$model name$
If \( <\text{model name}> \) is present, four things change:

- The mutual coupling inductor becomes a nonlinear, magnetic core device. The magnetic core’s B-H characteristics are analyzed using the Jiles-Atherton model (see Inductor coupling: Jiles-Atherton model).
- The inductors become windings, so the number specifying inductance now specifies the number of turns.
- The list of coupled inductors could be just one inductor.
- A model statement is required to specify the model parameters.

[size value]
Has a default value of 1.0 and scales the magnetic cross-section. It is intended to represent the number of lamination layers, so only one model statement is needed for each lamination type. For example:

\[ \begin{align*}
  & L1 5 9 20 ; \text{inductor having 20 turns} \\
  & K1 L1 1 K528T500_3C8; \text{Ferroxcube toroid core} \\
  & L2 3 8 15 ; \text{primary winding having} \\
  & \quad ; \text{15 turns} \\
  & L3 4 6 45 ; \text{secondary winding having} \\
  & \quad ; \text{45 turns} \\
  & K2 L2 L3 1 K528T500_3C8; \text{another core (not the same as K1)}
\end{align*}\]
Here is a Probe B-H display of 3C8 ferrite (Ferroxcube).

**Comments**

The linear branch relation for transient analysis is

\[ V_i = L_i \frac{dI_i}{dt} + M_{ij} \frac{dI_j}{dt} + M_{ik} \frac{dI_k}{dt} + \ldots \]

For U.C. Berkeley SPICE2: if there are several coils on a transformer, then there must be \( K \) statements coupling all combinations of inductor pairs. For instance, a transformer using a center-tapped primary and two secondaries could be written:

* PRIMARY
  L1 1 2 10uH
  L2 2 3 10uH
* SECONDARY
  L3 11 12 10uH
  L4 13 14 10uH
* MAGNETIC COUPLING
  K12 L1 L2 1
  K13 L1 L3 1
  K14 L1 L4 1
  K23 L2 L3 1
  K24 L2 L4 1
  K34 L3 L4 1

This older technique is still supported, but not required, for simulation. The same transformer can also be written:

* PRIMARY
  L1 1 2 10uH
  L2 2 3 10uH
* SECONDARY
  L3 11 12 10uH
  L4 13 14 10uH
* MAGNETIC COUPLING
  KALL L1 L2 L3 L4 1

Do not mix the two techniques.
The simulator uses the Jiles-Atherton model (see Inductor coupling: Jiles-Atherton model) to analyze the B-H curve of the magnetic core and calculate values for inductance and flux for each of the windings.

The state of the nonlinear core can be viewed in Probe by specifying $B(Kxxx)$ for the magnetization or $H(Kxxx)$ for the magnetizing influence. These values are not available for .PRINT (print) or .PLOT (plot) output.

### Capture parts

See your PSpice user’s guide for information about using nonlinear magnetic cores with transformers.

<table>
<thead>
<tr>
<th>Part name</th>
<th>Model type</th>
<th>Property</th>
<th>Property description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XFRM_LINEAR</td>
<td>transformer</td>
<td>L1_VALUE</td>
<td>winding inductances in Henries</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L2_VALUE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>COUPLING</td>
<td>coefficient of mutual coupling (must lie between 0 and 1)</td>
</tr>
<tr>
<td>K_LINEAR</td>
<td>transformer</td>
<td>Ln</td>
<td>inductor reference designator</td>
</tr>
<tr>
<td>XFRM_NONLINEAR</td>
<td>transformer</td>
<td>L1_TURNS</td>
<td>number of turns on each winding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L2_TURNS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>COUPLING</td>
<td>coefficient of mutual coupling (must lie between 0 and 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MODEL</td>
<td>nonlinear CORE model name</td>
</tr>
</tbody>
</table>

### Breakout parts

For non-stock passive and semiconductor devices, Capture provides a set of breakout parts designed for customizing model parameters for simulation. These are useful for setting up Monte Carlo and worst-case analyses with device and/or lot tolerances specified for individual model parameters. Another approach is to use the model editor to derive an instance model and customize this. For example, you could add device and/or lot tolerances to model parameters.

Basic breakout part names consist of the intrinsic PSpice A/D device letter plus the suffix BREAK. By default, the model name is the same as the part name and references the appropriate device model with all parameters set at their default. For instance, the DBREAK part references the DBREAK model which is derived from the intrinsic PSpice A/D D model (.MODEL DBREAK D)

### Using the KBREAK part

The inductor coupling part, KBREAK, can be used to couple up to six independent inductors on a schematic. A MODEL property is provided for using nonlinear magnetic core (CORE)
models, if desired. By default, KBREAK references the KBREAK model contained in breakout.lib; this model, in turn, uses the default CORE model parameters.

<table>
<thead>
<tr>
<th>Device type</th>
<th>Part name</th>
<th>Part library</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inductor coupling</td>
<td>KBREAK</td>
<td>BREAKOUT.OLB</td>
<td>COUPLING</td>
<td>coupling factor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Li</td>
<td>inductor reference</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>designator</td>
</tr>
</tbody>
</table>

The KBREAK part can be used to:

- Provide linear coupling between inductors.
- Reference a CORE model in a configured model library file.
- Define a user-defined CORE model with custom model parameter values.

The dot convention for the coupling is related to the direction in which the inductors are connected. The dot is always next to the first pin to be netlisted. For example, when the inductor part L is placed without rotation, the dotted pin is the left one. Rotate on the Edit menu (Ctrl+R) rotates the inductor +90°, making this pin the bottom pin.

Nonlinear coupling is not included in PSpice A/D Basics+.

For nonlinear coupling  
L1 must have a value; the rest may be left blank. The model must reference a CORE model such as those contained in MAGNETIC.LIB or other user-defined models. VALUE is set to the number of windings.

For linear coupling  
L1 and at least one other Li must have values; the rest may be left blank. The model reference must be blank. VALUE must be in Henries.
### Inductor coupling: Jiles-Atherton model

The Jiles-Atherton model is based on existing ideas of domain wall motion, including flexing and translation. The model derives an anhysteric magnetization curve by using a mean field technique, in which any domain is coupled to the magnetic field \((H)\) and the bulk magnetization \((M)\). This anhysteric value is the magnetization that would be reached in the absence of domain wall pinning. Hysteresis is modeled by the effects of pinning of domain walls on material defect sites. This impedance to motion and flexing due to the differential field exhibits all of the main features of real, nonlinear magnetic devices, such as the initial magnetization curve (initial permeability), saturation of magnetization, coercivity, remanence, and hysteresis loss.

A magnetic material that is comprised of loosely coupled domains has an equilibrium B-H curve, called the anhysteric. This curve is the locus of B-H values generated by superimposing a DC magnetic bias and a large AC signal that decays to zero. It is the curve representing minimum energy for the domains and is modeled, in theory, by

\[
M_{\text{an}} = MS \cdot H / ( |H| + A )
\]

where

- \(M_{\text{an}}\) = the anhysteric magnetization
- \(MS\) = the saturation magnetization
- \(H\) = the magnetizing influence (after GAP correction)
- \(A\) = a thermal energy parameter

For a given magnetizing influence \((H)\), the anhysteric magnetization is the global flux level the material would attain if the domain walls could move freely. The walls, however, are stopped or pinned on dislocations in the material. The wall remains pinned until enough magnetic potential is available to break free, and travel to the next pinning site. The theory...
supposes a mean energy required, per volume, to move domain walls. This is analogous to mechanical drag. A simplified equation of this is

\[
\text{change-in-magnetization} = \frac{\text{potential}}{\text{drag}}
\]

The irreversible domain wall motion can, therefore, be expressed as

\[
d_{\text{Mirr}}/dH = (M_{an} - M)/K
\]

where \(K\) is the pinning energy per volume (drag).

Reversible wall motion comes from flexing in the domain walls, especially when it is pinned at a dislocation due to the magnetic potential (that is, the magnetization is not the anhysteric value).

The theory supposes spherical flexure to calculate energy values and arrives at the (simplified) equation:

\[
d_{\text{Mrev}}/dH = C \cdot d(M_{an} - M)/dH
\]

where \(C\) is the domain flexing parameter.

The equation for the total magnetization is the sum of these two state equations:

\[
dM/dH = \left(\frac{1}{1 + C}\right) \cdot (M_{an} - M)/K + \left(\frac{C}{1 + C}\right) \cdot dM_{an}/dH
\]

Including air-gap effects in the inductor coupling model

If the gap thickness is small compared with the other dimensions of the core, you can assume that all of the magnetic flux lines go through the gap directly and that there is little fringing flux (having a modest amount of fringing flux only increases the effective air-gap length). Checking the field values around the entire magnetic path gives the equation:

\[
H_{\text{core}} \cdot L_{\text{core}} + H_{\text{gap}} \cdot L_{\text{gap}} = n \cdot I
\]

where \(n \cdot I\) is the sum of the amp-turns of the windings on the core. Also, the magnetization in the air-gap is negligible, so that \(B_{\text{gap}} = H_{\text{gap}}\) and \(B_{\text{gap}} = B_{\text{core}}\). These combine in the previous equation to yield:

\[
H_{\text{core}} \cdot L_{\text{core}} + B_{\text{core}} \cdot L_{\text{gap}} = n \cdot I
\]

This is a difficult equation to solve, especially for the Jiles-Atherton model, which is a state equation model rather than an explicit function (which one would expect, because the B-H curve depends on the history of the material). However, there is a graphical technique that solves for \(B_{\text{core}}\) and \(H_{\text{core}}\), given \(n \cdot I\), which is to:

1. Take the non-gapped B-H curve.
2. Extend a line from the current value of \(n \cdot I\) having a slope of \(L_{\text{core}}/L_{\text{gap}}\) (this would be vertical if \(L_{\text{gap}} = 0\)).
3. Find the intersection of the line using the B-H curve.

The intersection is the value for \(B_{\text{core}}\) and \(H_{\text{core}}\) for the \(n \cdot I\) of the gapped core. The \(n \cdot I\) value is the apparent or external value of \(H_{\text{core}}\), but the real value of \(H_{\text{core}}\) is less. The result is a smaller value for \(B_{\text{core}}\) and for the sheared-over B-H curves of a gapped core. The simulator implements the numerical equivalent of this graphical technique.

The resulting B-H values are recorded in the Probe data file as \(B_{\text{core}}\) and \(H_{\text{apparent}}\).
Getting core inductor coupling model values

Characterizing core materials can be performed using Parts, and verified by using PSpice and Probe. The model uses MKS (metric) units, however the results for Probe are converted to Gauss and Oersted, which can be displayed using $B(Kxxx)$ and $H(Kxxx)$. The traditional B-H curve is made by a transient run, ramping current through a test inductor, then displaying $B(Kxxx)$ and setting the X axis to $H(Kxxx)$.

For more information on the Jiles-Atherton model, see Reference [1] of References.

Transmission line coupling

If a K device is used to couple two transmission lines, then two coupling parameters are required.

<table>
<thead>
<tr>
<th>Device</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_m$</td>
<td>capacitive coupling</td>
<td>farad/length*</td>
<td>none</td>
</tr>
<tr>
<td>$L_m$</td>
<td>inductive coupling</td>
<td>henries/length*</td>
<td>none</td>
</tr>
</tbody>
</table>

* Length units must be consistent using the LEN parameter for the transmission lines being coupled.

These parameters can be thought of as the off-diagonal terms of a capacitive coupling matrix, $[C]$, and an inductive coupling matrix, $[L]$, respectively. $[C]$ and $[L]$ are both symmetric matrices, and for two coupled lines, the following relationships hold:

$$[C] = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \quad [L] = \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix}$$

$C_{12}$ represents the charge induced on the first conductor when the second conductor has a potential of one volt. In general, for a system of $N$ coupled lines, $C_{ij}$ is the charge on the $i^{th}$ conductor when the $j^{th}$ conductor is set to one volt, and all other conductors are grounded. The diagonal of the matrix is determined with the understanding that the self-capacitance is really the capacitance between the conductor and ground, so that:

$$C_{ii} = C_{ig} + \sum |C_{ij}|$$

where $C_{ig}$ is equal to the capacitance per unit length for the $i^{th}$ transmission line, and is provided along with the T device that describes the $i^{th}$ line. The simulator calculates $C_{ii}$ from this.

The values of $C_{ij}$ in the matrix are negative values. Note that the simulator assigns $-|C_m|$ to the appropriate $C_{ij}$, so that the sign used when specifying $C_m$ is ignored.

$L_{12}$ is defined in terms of the flux between the $1^{st}$ conductor and the ground plane, when the $2^{nd}$ conductor carries a current of one ampere. If there are more than two conductors, all other conductors are assumed to be open.

$L_{11}$ is equal to the inductance per unit length for the $1^{st}$ line and is obtained directly from the appropriate T device.
Example

The following circuit fragment shows an example using two coupled lines:

T1 1 0 2 0 R=.31 L=.38u G=6.3u C=70p LEN=1
T2 3 0 4 0 R=.29 L=.33u G=6.0u C=65p LEN=1
K12 T1 T2 Lm=.04u Cm=6p

This fragment leads to the following [C] and [L]:

\[
\begin{bmatrix}
76p & -6p \\
-6p & 71p
\end{bmatrix}
\begin{bmatrix}
0.38u & 0.04u \\
0.04u & 0.33u
\end{bmatrix}
\]

The model used to simulate this system is based on the approach described by Tripathi and Rettig in Reference [1] of References and is extended for lossy lines by Roychowdhury and Pederson in Reference [2]. The approach involves computing the system propagation modes by extracting the eigenvalues and eigenvectors of the matrix product [L][C].

This model is not general for lossy lines.

Lossy lines

For the lossy line case, the matrix product to be decoupled is actually:

\[(R+sl)(G+sC)\]

where:

\[s\] = the Laplace variable
\[R\] = the resistance per unit length matrix
\[G\] = the conductance per unit length matrix.

The modes obtained from [L][C] represent a high frequency asymptote for this system. Simulation results should be good approximations for low-loss lines. However, as shown in reference [2], the approximation becomes exact for homogeneous, equally-spaced lossy lines, provided that coupling beyond immediately adjacent lines is negligible (i.e., the coupling matrices are tridiagonal and Toeplitz).

Coupled ideal lines can be modeled by setting R and G to zero. The Z0/TD parameter set is not supported for coupled lines.
References

For a further description of the Jiles-Atherton model, refer to:


For more information on transmission line coupling, refer to:


**Inductor**

**General form**

L<name> <(+ node>) <(- node)> [model name] <value>  
+ [IC=<initial value>]

**Examples**

LLOAD 15 0 20mH  
L2 1 2 .2E-6  
LCHOKE 3 42 LMOD .03  
LSENSE 5 12 2UH IC=2mA

**Model form**

.MODEL <model name> IND [model parameters]

---

**Arguments and options**

- **(+ and -) nodes**
  - Define the polarity when the inductor has a positive voltage across it.
  
  The first node listed (or pin one in Capture), is defined as positive. The voltage across the component is therefore defined as the first node voltage less the second node voltage.

  Positive current flows from the (+) node through the inductor to the (-) node. Current flow from the first node through the component to the second node is considered positive.

- **[model name]**
  - If [model name] is left out, then the effective value is <value>.
  
  If [model name] is specified, then the effective value is given by the model parameters; see **Inductance value formula**.

  If the inductor is associated with a Core model, then the effective value is the number of turns on the core. Otherwise, the effective value is the inductance. See the Model Form statement for the K device in **Inductor coupling (and magnetic core)** for more information on the Core model.

- **<initial value>**
  - Is the initial current through the inductor during the bias point calculation.

  It can also be specified in a circuit file using a .IC statement as follows:

  `.IC I(L<name>) <initial value>`

  For details on using the .IC statement in a circuit file, see **.IC (initial bias point condition)** and refer to your PSpice user’s guide for more information.
Capture parts

For standard L parts, the effective value of the part is set directly by the VALUE property. In general, inductors should have positive component values (VALUE property). In all cases, components must not be given a value of zero.

However, there are cases when negative component values are desired. This occurs most often in filter designs that analyze an RLC circuit equivalent to a real circuit. When transforming from the real to the RLC equivalent, it is possible to end up with negative component values.

PSpice A/D allows negative component values for bias point, DC sweep, AC, and noise analyses. A transient analysis may fail for a circuit with negative components. Negative inductors may create instabilities in time that the analysis cannot handle.

<table>
<thead>
<tr>
<th>Part name</th>
<th>Model type</th>
<th>Property</th>
<th>Property description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>inductor</td>
<td>VALUE</td>
<td>inductance</td>
</tr>
<tr>
<td>IC</td>
<td></td>
<td></td>
<td>initial current</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>through the inductor</td>
</tr>
<tr>
<td>XFRM_LINEAR</td>
<td>transformer</td>
<td>L1_VALUE</td>
<td>winding inductances</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>in Henries</td>
</tr>
<tr>
<td>L2_VALUE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUPLING</td>
<td></td>
<td></td>
<td>coefficient of</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>mutual coupling (must</td>
</tr>
<tr>
<td>K_LINEAR</td>
<td>transformer</td>
<td>Ln</td>
<td>inductor reference</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>designator</td>
</tr>
</tbody>
</table>
Breakout parts

For non-stock passive and semiconductor devices, Capture provides a set of breakout parts designed for customizing model parameters for simulation. These are useful for setting up Monte Carlo and worst-case analyses with device and/or lot tolerances specified for individual model parameters. Another approach is to use the model editor to derive an instance model and customize this. For example, you could add device and/or lot tolerances to model parameters.

Basic breakout part names consist of the intrinsic PSpice A/D device letter plus the suffix BREAK. By default, the model name is the same as the part name and references the appropriate device model with all parameters set at their default. For instance, the DBREAK part references the DBREAK model, which is derived from the intrinsic PSpice A/D D model (.MODEL DBREAK D).

For breakout part LBREAK, the effective value is computed from a formula that is a function of the specified VALUE property.

<table>
<thead>
<tr>
<th>Device type</th>
<th>Part name</th>
<th>Part library file</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inductor</td>
<td>LBREAK</td>
<td>BREAKOUT.OLB</td>
<td>VALUE</td>
<td>inductance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IC</td>
<td>initial current through the inductor during bias point calculation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MODEL</td>
<td>IND model name</td>
</tr>
</tbody>
</table>
Inductor model parameters

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>Inductance multiplier</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>IL1</td>
<td>Linear current coefficient</td>
<td>amp</td>
<td>0.0</td>
</tr>
<tr>
<td>IL2</td>
<td>Quadratic current coefficient</td>
<td>amp²</td>
<td>0.0</td>
</tr>
<tr>
<td>TC1</td>
<td>Linear temperature coefficient</td>
<td>°C⁻¹</td>
<td>0.0</td>
</tr>
<tr>
<td>TC2</td>
<td>Quadratic temperature coefficient</td>
<td>°C⁻²</td>
<td>0.0</td>
</tr>
<tr>
<td>T_ABS</td>
<td>Absolute temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_MEASURED</td>
<td>Measured temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_REL_GLOBAL</td>
<td>Relative to current temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_REL_LOCAL</td>
<td>Relative to AKO model temperature</td>
<td>°C</td>
<td></td>
</tr>
</tbody>
</table>

* For information on T_MEASURED, T_ABS, T_REL_GLOBAL, and T_REL_LOCAL, see MODEL (model definition).

Inductor equations

Inductance value formula

If [model name] is specified, then the effective value is given by:

<value>·\(L\cdot(1+IL1\cdot I^1+IL2\cdot I^2)\cdot(1+TC1\cdot(T-Tnom)+TC2\cdot(T-Tnom)^2)\)

where <value> is normally positive (though it can be negative, but not zero). \(T_{nom}\) is the nominal temperature (set using TNOM option).

Inductor equation for noise

The inductor does not have a noise model.
MOSFET

General form

M<name> <drain node> <gate node> <source node>
+ <bulk/substrate node> <model name>
+ [L=<value>] [W=<value>]
+ [AD=<value>] [AS=<value>]
+ [PD=<value>] [PS=<value>]
+ [NRD=<value>] [NRS=<value>]
+ [NRG=<value>] [NRB=<value>]
+ [M=<value>] [N=<value>]

Examples

M1 14 2 13 0 PNOM L=25u W=12u
M13 15 3 0 0 PSTRONG
M16 17 3 0 0 PSTRONG M=2
M28 0 2 100 100 NWEAK L=33u W=12u
+ AD=288p AS=288p PD=60u PS=60u NRD=14 NRS=24 NRG=10

Model form

.MODEL <model name> NMOS [model parameters]
.MODEL <model name> PMOS [model parameters]

Description

The MOSFET is modeled as an intrinsic MOSFET using ohmic resistances in series with the drain, source, gate, and bulk (substrate). There is also a shunt resistance (RDS) in parallel with the drain-source channel.

Arguments and options

L and W
are the channel length and width, which are decreased to get the effective channel length and width. They can be specified in the device, .MODEL (model definition), or .OPTIONS (analysis options) statements. The value in the device statement supersedes the value in the model statement, which supersedes the value in the .OPTIONS statement. Defaults for L and W can be set in the .OPTIONS statement. If L or W defaults are not set, their default value is 100 u.

[L=<value>] [W=<value>] cannot be used in conjunction with Monte Carlo analysis.
AD and AS
The drain and source diffusion areas. Defaults for AD and AS can be set in the .OPTIONS statement. If AD or AS defaults are not set, their default value is 0.

PD and PS
The drain and source diffusion perimeters. Their default value is 0.

JS
Can specify the drain-bulk and source-bulk saturation currents. JS is multiplied by AD and AS.

IS
Can also specify the drain-bulk and source-bulk saturation currents. IS is an absolute value.

CJ
Can specify the zero-bias depletion capacitances. CJ is multiplied by AD and AS.

CJSW
Can also specify the zero-bias depletion capacitances. CJSW is multiplied by PD and PS.

CBD and CBS
Can also specify the zero-bias depletion capacitances. CBD and CBS are absolute values.

NRD, NRS, NRG, and NRB
Multipliers (in units of squares) that can be multiplied by RSH to yield the parasitic (ohmic) resistances of the drain (RD), source (RS), gate (RG), and substrate (RB), respectively. NRD, NRS, NRG, and NRB default to 0.

Consider a square sheet of resistive material. Analysis shows that the resistance between two parallel edges of such a sheet depends upon its composition and thickness, but is independent of its size as long as it is square. In other words, the resistance will be the same whether the square’s edge is 2 mm, 2 cm, or 2 m. For this reason, the sheet resistance of such a layer, abbreviated RSH, has units of ohms per square.

M (NP)
A parallel device multiplier (default = 1), which simulates the effect of multiple devices in parallel. (NP is an alias for M.)

The effective width, overlap and junction capacitances, and junction currents of the MOSFET are multiplied by M. The parasitic resistance values (e.g., RD and RS) are divided by M. Note the third example: it shows a device twice the size of the second example.
N (NS)  
A series device multiplier (default value = 1.0) for the Level 5 model only, which simulates an approximation of the effect of multiple devices in series. NS is an aliased name for N.

There are some things to keep in mind while using this parameter. The parameter N is used to derive the effective length, \( \text{Leff} = N \cdot (L+DL) \), of a transistor drawn as N elements of width W and length L in series (in other words, the drain of element [K] is the source of element [K+1], and the gates are tied together). The short-channel effects included in the pinch-off voltage calculation, however, are evaluated using the effective length L+DL of each element. Except for this, everything is calculated as if the transistor were laid out as a single element of length \( \text{L} = \text{Leff} - DL = N \cdot (L+DL) - DL \).

In this compact formulation, the intermediate drain/source diffusions appearing along the channel are ignored (that is, junction capacitance and diffusion resistances are assumed to be zero). As a consequence, DC, AC and transient analyses can yield different results compared with the standard device declaration, particularly at higher frequencies. A closer match is obtained for long devices, or devices with low RS and RD and high UCRIT. Be sure to evaluate the accuracy of this compact formulation and to check the validity of the underlying approximations.

Comments  
The simulator provides six MOSFET device models, which differ in the formulation of the I-V characteristic. The \textbf{LEVEL} parameter selects among different models as shown below. For more information, see References.

- **LEVEL=1**  Shichman-Hodges model (see reference [1])
- **LEVEL=2**  geometry-based, analytic model (see reference [2])
- **LEVEL=3**  semi-empirical, short-channel model (see reference [2])
- **LEVEL=4**  BSIM model (see reference [3])
- **LEVEL=5**  EKV model version 2.6 (see reference [10])
- **LEVEL=6**  BSIM3 model version 2.0 (see reference [7])
- **LEVEL=7**  BSIM3 model version 3.1 (see reference [8])
Capture parts

The following table lists the set of MOSFET breakout parts designed for customizing model parameters for simulation. These are useful for setting up Monte Carlo and worst-case analyses with device and/or lot tolerances specified for individual model parameters.

<table>
<thead>
<tr>
<th>Part name</th>
<th>Model type</th>
<th>Property</th>
<th>Property description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBREAKN</td>
<td>NMOS</td>
<td>L</td>
<td>channel length</td>
</tr>
<tr>
<td>MBREAKN3</td>
<td></td>
<td>W</td>
<td>channel width</td>
</tr>
<tr>
<td>MBREAKN4</td>
<td></td>
<td>AD</td>
<td>drain diffusion area</td>
</tr>
<tr>
<td>MBREAKP</td>
<td>PMOS</td>
<td>AS</td>
<td>source diffusion area</td>
</tr>
<tr>
<td>MBREAKP3</td>
<td></td>
<td>PD</td>
<td>drain diffusion perimeter</td>
</tr>
<tr>
<td>MBREAKP4</td>
<td></td>
<td>PS</td>
<td>source diffusion perimeter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NRD</td>
<td>relative drain resistivity (in squares)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NRS</td>
<td>relative source resistivity (in squares)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NRG</td>
<td>relative gate resistivity (in squares)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NRB</td>
<td>relative substrate resistivity (in squares)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M</td>
<td>device multiplier (simulating parallel devices)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MODEL</td>
<td>NMOS or PMOS model name</td>
</tr>
</tbody>
</table>

Setting operating temperature

Setting operating temperature can be set to be different from the global circuit temperature by defining one of the model parameters: T_ABS, T_REL_GLOBAL, or T_REL_LOCAL. Additionally, model parameters can be assigned unique measurement temperatures using the T_MEASURED model parameter. For more information, see MOSFET model parameters.
MOSFET model parameters

For all model levels

The parameters common to all model levels are primarily parasitic element values such as series resistance, overlap and junction capacitance, and so on.

Model levels 1, 2, and 3

The DC characteristics of the first three model levels are defined by the parameters VTO, KP, LAMBDA, PHI, and GAMMA. These are computed by the simulator if process parameters (e.g., TOX, and NSUB) are given, but the user-specified values always override. VTO is positive (negative) for enhancement mode and negative (positive) for depletion mode of N-channel (P-channel) devices.

The default value for TOX is 0.1 µ for Levels 2 and 3, but is unspecified for Level 1, which discontinues the use of process parameters.

For MOSFETs the capacitance model has been changed to conserve charge, affecting only the Level 1, 2, and 3 models.

Effective length and width for device parameters are calculated with the formula:

\[ P_i = P_0 + P_L/L_e + P_W/W_e \]

where:

\[ L_e = \text{effective length} = L - (L_D \cdot 2) \]
\[ W_e = \text{effective width} = W - (W_D \cdot 2) \]

See .MODEL (model definition) for more information.

Model level 4

Unlike the other models in PSpice, the BSIM model is designed for use with a process characterization system that provides all parameters. Therefore, there are no defaults specified for the parameters, and leaving one out can cause problems.

The LEVEL=4 (BSIM1) model parameters are all values obtained from process characterization, and can be generated automatically. Reference [4] of References describes a means of generating a process file, which must then be converted into .MODEL (model definition) statements for inclusion in the Model Library or circuit file. (The simulator does not read process files.)

The level 4 (BSIM) and level 6 (BSIM3 version 2) models have their own capacitance model, which conserves charge and remains unchanged. References [6] and [7] describe the equations for the capacitance due to channel charge.

In the following MOSFET model parameters list, parameters marked with a ζ in the Default column also have corresponding parameters with a length and width dependency. For
example, VFB is a basic parameter using units of volts, and LVFB and WVFB also exist and have units of volt·µ. The formula
\[ P_i = P_0 + P_L/L_e + P_w/W_e \]
is used to evaluate the parameter for the actual device, where:

\[ L_e = \text{effective length} = L - DL \]
\[ W_e = \text{effective width} = W - DW \]

**Model level 5 (EKV version 2.6)**

The EKV model is a scalable and compact model built on fundamental physical properties of the device. Use this model to design low-voltage, low-current analog, and mixed analog-digital circuits that use sub-micron technologies. The charge-based static, quasi-static dynamic, and noise models are all derived from the normalized transconductance-to-current ratio, which is accurately described for all levels of current, including the moderate inversion region. A single I-V expression preserves the continuity of first- and higher-order derivatives with respect to any terminal voltage in all regions of device operation.

Version 2.6 models the following:

- geometrical and process related aspects of the device (oxide thickness, junction depth, effective channel length and width, and so on)
- effects of doping profile and substrate effects
- weak, moderate, and strong inversion behavior
- mobility effects due to vertical and lateral fields and carrier velocity saturation
- short-channel effects such as channel-length modulation, source and drain charge sharing, and the reverse short channel effect
- thermal and flicker noise modeling
- short-distance geometry and bias-dependent device matching for Monte Carlo analysis.

For more detailed model information, see reference [10] of References.

**Additional notes**

**Note 1** The DL and DW parameters usually have a negative value.

**Note 2** 0 (zero) and O (the letter O) are not interchangeable. For example, use VTO, not VT0 (VTO is referenced to the bulk); use E0, not EO; use Q0, not QO.

**Note 3** Use the AVTO, AKP, and AGAMMA model parameters with a DEV tolerance to perform Monte Carlo and Sensitivity/Worst-Case analyses. Their default values cannot be changed.

The device-to-device matching of MOSFETs depends on the gate area, W · L. Using AVTO, AKP, and AGAMMA with a DEV tolerance applies the matching scaling law for the model equations and derives the device matching statistics (DEV tolerance) from a single normalized parameter. (Without these parameters, you would need to use a dedicated .MODEL card with a DEV tolerance for VTO, KP and GAMMA for each value of the gate area used in your design.)
Do not apply the LOT specification, which is a measure of the ability of the process to control the absolute value of a model parameter, to $AVTO$, $AKP$, and $AGAMMA$, because this would be redundant with the LOT specification for $VTO$, $KP$, and $GAMMA$.

**Note 4** Use the model parameter $HDIF$ with the device parallel multiplier, $M$, to set default values for $AD$, $AS$, $PD$, and $PS$. Use $HDIF$ only for the MOSEKV (Level 5) model. When $HDIF$ is specified, the following equations are used.

\[
\begin{align*}
NRD &= HDIF/W \\
NRS &= HDIF/W \\
\end{align*}
\]

For $M = 1$, the following equations are used.

\[
\begin{align*}
AD &= (2 \cdot HDIF) \cdot W \\
AS &= (2 \cdot HDIF) \cdot W \\
PD &= 2 \cdot ((2 \cdot HDIF) + W) \\
PS &= 2 \cdot (2 \cdot HDIF) + W \\
\end{align*}
\]

For $M \geq 2$ and even:

\[
\begin{align*}
AD &= HDIF \cdot W \\
AS &= (HDIF + (2 \cdot HDIF)/M) \cdot W \\
PD &= (2 \cdot HDIF) + W \\
PS &= (2 \cdot HDIF) + W + 2 \cdot ((2 \cdot HDIF) + W)/M \\
\end{align*}
\]

For $M \geq 2$ and odd:

\[
\begin{align*}
AD &= (HDIF + (HDIF/M)) \cdot W \\
AS &= (HDIF + (HDIF/M)) \cdot W \\
PD &= (2 \cdot HDIF) + W + ((2 \cdot HDIF) + W)/M \\
PS &= (2 \cdot HDIF) + W + ((2 \cdot HDIF) + W)/M \\
\end{align*}
\]

**Note 5** If RSHG is specified, the default value for $NRG$ is set to $0.5 \cdot W/L$.

**Note 6** The model parameters $TOX$, $NSUB$, $VFB$, $UO$, and $VMAX$ accommodate scaling behavior of the process and basic intrinsic model parameters, as well as statistical circuit simulation. These parameters are only used if $COX$, $GAMMA$, and/or $PHI$, $VTO$, $KP$, and $UCRIT$ are not specified, respectively. Furthermore, a simpler mobility reduction model due to vertical field is accessible through the mobility reduction coefficient, $THETA$. $THETA$ is only used if $E0$ is not specified.
Model level 6 (BSIM3 version 2.0)

The Level 6 Advanced parameters should not be changed unless the detail structure of the device is known and has specific, meaningful values.

The BSIM3 model is a physical model using extensive built-in dependencies of important dimensional and processing parameters. It includes the major effects that are important to modeling deep-submicrometer MOSFETs, such as threshold voltage reduction, nonuniform doping, mobility reduction due to the vertical field, bulk charge effect, carrier velocity saturation, drain-induced barrier lowering (DIBL), channel length modulation (CLM), hot-carrier-induced output resistance reduction, subthreshold conduction, source/drain parasitic resistance, substrate current induced body effect (SCBE), and drain voltage reduction in LDD structure. For additional, detailed model information, see References.

Additional notes

Note 1  If any of the following BSIM3 version 2.0 model parameters are not explicitly specified, they are calculated using the following equations.

\[
V_{TH0} = V_{FB} + \Phi + K_1 \frac{\Phi}{V_{BM}}
\]

\[
K_1 = \Gamma_{M1} - 2 \cdot K_2 \cdot \frac{\Phi}{V_{BM}}
\]

\[
K_2 = \frac{\Gamma_{M0} - \Gamma_{M1}}{2 \cdot \frac{\Phi}{V_{BM}}}
\]

\[
V_{BF} = V_{TH0} - \frac{\Phi}{K_1} \cdot \frac{\Phi}{V_{BM}}
\]

\[
\Phi = 2 V_m \ln \left( \frac{N_{PEAK}}{n_i} \right)
\]

\[
\Gamma_{M1} = \frac{2q \varepsilon_{Si} N_{PEAK}}{\varepsilon_{Si} COX}
\]

\[
\Gamma_{M2} = \frac{2q \varepsilon_{Si} N_{SUB}}{\varepsilon_{Si} COX}
\]

\[
V_{BX} = \Phi - q \cdot N_{PEAK} \cdot X_T^2 / (2 \varepsilon_{Si})
\]

\[
L_{ITL} = \frac{\varepsilon_{Si} TOX}{\varepsilon_{OX}}
\]

Note 2  Default values listed for the BSIM3 version 2.0 parameters \textbf{UA}, \textbf{UB}, \textbf{UC}, \textbf{UA1}, \textbf{AB1}, and \textbf{UC1} are used for simplified mobility modeling.

Model level 7 (BSIM3 version 3.1)

The BSIM3 version 3.1 model was developed by the University of California, Berkeley, as a deep submicron MOSFET model with the same physical basis as the BSIM3 version 2 model, but with a number of major enhancements, such as a single I-V expression to describe current and output conductance in all regions of device operation, better modeling of narrow width devices, a reformulated capacitance model to improve short and narrow geometry models, a
new relaxation time model to improve transient modeling, and improved model fitting of various W/L ratios using one parameter set. BSIM3 version 3.1 retains the extensive built-in dependencies of dimensional and processing parameters of BSIM3 version 2. For additional, detailed model information, see Reference [8] of References.

Additional notes

Note 1  If any of the following BSIM3 version 3.1 model parameters are not explicitly specified, they are calculated using the following equations:

If VTHO is not specified, then:

\[ VTHO = VFB + \phi_s K1/\sqrt{\phi_s} \]

where:

\[ VFB = -1.0 \]

If VTHO is specified, then:

\[ VFB = VTHO - \phi_s + K1/\sqrt{\phi_s} \]

\[ VBX = \phi_s - q \cdot NCH \cdot XT^2 / 2 \cdot \varepsilon_s \]

\[ CF = \left( \frac{2\varepsilon_{ox}}{\pi} \right) \ln \left( 1 + 4 \times 10^{-7} \right) \]

where

\[ E_g(T) = \text{the energy bandgap at temperature } T = 1.16 \left( \frac{7.02 \cdot 10^{-4} \cdot T^2}{T + 1108} \right) \]

Note 2  If K1 AND K2 are not specified, they are calculated using the following equations:

\[ K1 = \text{GAMMA2} - 2K2 \cdot \phi_s / \sqrt{\phi_s} - \text{VBM} \]

\[ K2 = \frac{\text{GAMMA1} - \text{GAMMA2} \cdot \sqrt{\phi_s - \text{VBM} - \phi_s / \sqrt{\phi_s} + \text{VBM}}}{2 \cdot \phi_s \cdot \sqrt{\phi_s - \text{VBM} - \phi_s / \sqrt{\phi_s} + \text{VBM}}} \]

where:

\[ \phi_s = 2Vt \cdot \ln \left( \frac{\text{NCH}}{n_i} \right) \]

\[ Vt = \frac{k \cdot T}{q} \]

\[ n_i = 1.45 \cdot 10^{10} \left( \frac{T}{300.15} \right)^{1.5} \cdot \exp \left( 21.5565981 - \frac{E_g(T)}{2Vt} \right) \]

Note 3  If NCH is not given and GAMMA1 is given, then:

\[ NCH = \frac{\text{GAMMA1}^2 \cdot (Cox)^2}{2q \cdot \varepsilon_{si}^2} \]

If neither GAMMA1 nor NCH is given, then NCH has a default value of 1.7e23 1/m^3 and GAMMA1 is calculated from NCH:

\[ \text{GAMMA1} = \frac{\sqrt{2q \cdot \varepsilon_{si} \cdot \text{NCH}}}{\text{Cox}} \]
If \( \text{GAMMA}_2 \) is not given, then:

\[
\text{GAMMA}_2 = \sqrt{\frac{2q \cdot \varepsilon_{si} \cdot \text{NSUB}}{\text{Cox}}}
\]

**Note 3**  
If \( \text{CGSO} \) is not given and \( \text{DLC} > 0 \), then:

\[
\text{CGSO} = (\text{DLC} \cdot \text{Cox}) - \text{CGSL}
\]

If the previously calculated \( \text{CGSO} < 0 \), then:

\[
\text{CGSO} = 0
\]

Else:

\[
\text{CGSO} = 0.6 \cdot XJ \cdot \text{Cox}
\]

**Note 4**  
If \( \text{CGDO} \) is not given and \( \text{DLC} > 0 \), then:

\[
\text{CGDO} = (\text{DLC} \cdot \text{Cox}) - \text{CGSL}
\]

If the previously calculated \( \text{CGDO} < 0 \), then

\[
\text{CGDO} = 0
\]

Else:

\[
\text{CGDO} = 0.6 \cdot XJ \cdot \text{Cox}
\]
### MOSFET model parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF</td>
<td>flicker noise exponent</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>CBD</td>
<td>zero-bias bulk-drain p-n capacitance</td>
<td>farad</td>
<td>0</td>
</tr>
<tr>
<td>CBS</td>
<td>zero-bias bulk-source p-n capacitance</td>
<td>farad</td>
<td>0</td>
</tr>
<tr>
<td>CGBO</td>
<td>gate-bulk overlap capacitance/channel length</td>
<td>farad/meter</td>
<td>0</td>
</tr>
<tr>
<td>CGDO</td>
<td>gate-drain overlap capacitance/channel width</td>
<td>farad/meter</td>
<td>0</td>
</tr>
<tr>
<td>CGSO</td>
<td>gate-source overlap capacitance/channel width</td>
<td>farad/meter</td>
<td>0</td>
</tr>
<tr>
<td>CJ</td>
<td>bulk p-n zero-bias bottom capacitance/area</td>
<td>farad/meter</td>
<td>0</td>
</tr>
<tr>
<td>CJSW</td>
<td>bulk p-n zero-bias sidewall capacitance/length</td>
<td>farad/meter</td>
<td>0</td>
</tr>
<tr>
<td>FC</td>
<td>bulk p-n forward-bias capacitance coefficient</td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>GDSNOI</td>
<td>channel shot noise coefficient (use with NLEV=3)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>IS</td>
<td>bulk p-n saturation current</td>
<td>amp</td>
<td>1E-14</td>
</tr>
<tr>
<td>JS</td>
<td>bulk p-n saturation current/area</td>
<td>amp/meter</td>
<td>0</td>
</tr>
<tr>
<td>JSSW</td>
<td>bulk p-n saturation sidewall current/length</td>
<td>amp/meter</td>
<td>0</td>
</tr>
<tr>
<td>KF</td>
<td>flicker noise coefficient</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>L</td>
<td>channel length</td>
<td>meter</td>
<td>DEFL</td>
</tr>
<tr>
<td>LEVEL</td>
<td>model index</td>
<td>meter</td>
<td>1</td>
</tr>
<tr>
<td>MJ</td>
<td>bulk p-n bottom grading coefficient</td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>MJSW</td>
<td>bulk p-n sidewall grading coefficient</td>
<td></td>
<td>0.33</td>
</tr>
<tr>
<td>N</td>
<td>bulk p-n emission coefficient</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>NLEV</td>
<td>noise equation selector</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>PB</td>
<td>bulk p-n bottom potential</td>
<td>volt</td>
<td>0.8</td>
</tr>
<tr>
<td>PBSW</td>
<td>bulk p-n sidewall potential</td>
<td>volt</td>
<td>PB</td>
</tr>
<tr>
<td>RB</td>
<td>bulk ohmic resistance</td>
<td>ohm</td>
<td>0</td>
</tr>
<tr>
<td>RD</td>
<td>drain ohmic resistance</td>
<td>ohm</td>
<td>0</td>
</tr>
<tr>
<td>RDS</td>
<td>drain-source shunt resistance</td>
<td>ohm</td>
<td>infinite</td>
</tr>
<tr>
<td>RG</td>
<td>gate ohmic resistance</td>
<td>ohm</td>
<td>0</td>
</tr>
<tr>
<td>RS</td>
<td>source ohmic resistance</td>
<td>ohm</td>
<td>0</td>
</tr>
<tr>
<td>RSH</td>
<td>drain, source diffusion sheet resistance</td>
<td>ohm/square</td>
<td>0</td>
</tr>
<tr>
<td>TT</td>
<td>bulk p-n transit time</td>
<td>sec</td>
<td>0</td>
</tr>
</tbody>
</table>
## MOSFET model parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_ABS †</td>
<td>absolute temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_MEASURED †</td>
<td>measured temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_REL_GLOBAL †</td>
<td>relative to current temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_REL_LOCAL †</td>
<td>relative to AKO model temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>channel width</td>
<td>meter</td>
<td>DEFW</td>
</tr>
</tbody>
</table>

### Levels 1, 2, and 3

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELTA</td>
<td>width effect on threshold</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>ETA</td>
<td>static feedback (Level 3)</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>GAMMA</td>
<td>bulk threshold parameter</td>
<td>volt(^{1/2})</td>
<td>see page 178</td>
</tr>
<tr>
<td>KP</td>
<td>transconductance coefficient</td>
<td>amp/volt(^2)</td>
<td>2.0E-5</td>
</tr>
<tr>
<td>KAPPA</td>
<td>saturation field factor (Level 3)</td>
<td></td>
<td>0.2</td>
</tr>
<tr>
<td>LAMBDA</td>
<td>channel-length modulation (Levels 1 and 2)</td>
<td>volt(^{-1})</td>
<td>0.0</td>
</tr>
<tr>
<td>LD</td>
<td>lateral diffusion (length)</td>
<td>meter</td>
<td>0.0</td>
</tr>
<tr>
<td>NEFF</td>
<td>channel charge coefficient (Level 2)</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>NFS</td>
<td>fast surface state density</td>
<td>1/cm(^2)</td>
<td>0.0</td>
</tr>
<tr>
<td>NSS</td>
<td>surface state density</td>
<td>1/cm(^2)</td>
<td>none</td>
</tr>
<tr>
<td>NSUB</td>
<td>substrate doping density</td>
<td>1/cm(^3)</td>
<td>none</td>
</tr>
<tr>
<td>PHI</td>
<td>surface potential</td>
<td>volt</td>
<td>0.6</td>
</tr>
<tr>
<td>THETA</td>
<td>mobility modulation (Level 3)</td>
<td>volt(^{-1})</td>
<td>0.0</td>
</tr>
<tr>
<td>TOX</td>
<td>oxide thickness</td>
<td>meter</td>
<td>see page 178</td>
</tr>
<tr>
<td>TPG</td>
<td>Gate material type:</td>
<td></td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>+1 = opposite of substrate</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1 = same as substrate</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = aluminum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UCRIT</td>
<td>mobility degradation critical field (Level 2)</td>
<td>volt/cm</td>
<td>1.0E4</td>
</tr>
<tr>
<td>UEXP</td>
<td>mobility degradation exponent (Level 2)</td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td>UTRA</td>
<td>(not used)</td>
<td>mobility degradation transverse field coefficient</td>
<td>0.0</td>
</tr>
<tr>
<td>UO</td>
<td>surface mobility (The second character is the letter O, not the numeral zero.)</td>
<td>cm(^2)/volt·sec</td>
<td>600</td>
</tr>
<tr>
<td>VMAX</td>
<td>maximum drift velocity</td>
<td>meter/sec</td>
<td>0</td>
</tr>
</tbody>
</table>
### MOSFET model parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VTO</strong></td>
<td>zero-bias threshold voltage</td>
<td>volt</td>
<td>0</td>
</tr>
<tr>
<td><strong>WD</strong></td>
<td>lateral diffusion (width)</td>
<td>meter</td>
<td>0</td>
</tr>
<tr>
<td><strong>XJ</strong></td>
<td>metallurgical junction depth (Levels 2 and 3)</td>
<td>meter</td>
<td>0</td>
</tr>
<tr>
<td><strong>XQC</strong></td>
<td>fraction of channel charge attributed to drain</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td><strong>DL</strong></td>
<td>Channel shortening</td>
<td>mu-m (1E-6*m)</td>
<td></td>
</tr>
<tr>
<td><strong>DW</strong></td>
<td>Channel narrowing</td>
<td>mu-m (1E-6*m)</td>
<td></td>
</tr>
<tr>
<td><strong>ETA</strong></td>
<td>Zero-bias drain-induced barrier lowering coefficient</td>
<td>ζ</td>
<td></td>
</tr>
<tr>
<td><strong>K1</strong></td>
<td>Body effect coefficient</td>
<td>volt$^{1/2}$</td>
<td>ζ</td>
</tr>
<tr>
<td><strong>K2</strong></td>
<td>Drain/source depletion charge sharing coefficient</td>
<td>ζ</td>
<td></td>
</tr>
<tr>
<td><strong>MUS</strong></td>
<td>Mobility at zero substrate bias and Vds=Vdd</td>
<td>cm$^2$/volt·sec</td>
<td>ζ</td>
</tr>
<tr>
<td><strong>MUZ</strong></td>
<td>Zero-bias mobility</td>
<td>cm$^2$/volt·sec</td>
<td></td>
</tr>
<tr>
<td><strong>N0</strong></td>
<td>Zero-bias subthreshold slope coefficient</td>
<td>ζ</td>
<td></td>
</tr>
<tr>
<td><strong>NB</strong></td>
<td>Sens. of subthreshold slope to substrate bias</td>
<td>ζ</td>
<td></td>
</tr>
<tr>
<td><strong>ND</strong></td>
<td>Sens. of subthreshold slope to drain bias</td>
<td>ζ</td>
<td></td>
</tr>
<tr>
<td><strong>PHI</strong></td>
<td>Surface inversion potential</td>
<td>volt</td>
<td>ζ</td>
</tr>
<tr>
<td><strong>TEMP</strong></td>
<td>Temperature at which parameters were measured</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td><strong>TOX</strong></td>
<td>Gate-oxide thickness</td>
<td>mu-m (1E-6*m)</td>
<td></td>
</tr>
<tr>
<td><strong>U0</strong></td>
<td>Zero-bias transverse-field mobility degradation</td>
<td>volt$^{-1}$</td>
<td>ζ</td>
</tr>
<tr>
<td><strong>U1</strong></td>
<td>Zero-bias velocity saturation</td>
<td>μ/volt</td>
<td>ζ</td>
</tr>
<tr>
<td><strong>VDD</strong></td>
<td>Measurement bias range</td>
<td>volts</td>
<td></td>
</tr>
<tr>
<td><strong>VFB</strong></td>
<td>Flat-band voltage</td>
<td>volt</td>
<td>ζ</td>
</tr>
<tr>
<td><strong>WDF</strong></td>
<td>Drain, source junction default width</td>
<td>meter</td>
<td></td>
</tr>
<tr>
<td><strong>X2E</strong></td>
<td>Sens. of drain-induced barrier lowering effect to substrate bias</td>
<td>volt$^{-1}$</td>
<td>ζ</td>
</tr>
<tr>
<td><strong>X2MS</strong></td>
<td>Sens. of mobility to substrate bias @ Vds=0</td>
<td>cm$^2$/volt$^2$-sec</td>
<td>ζ</td>
</tr>
<tr>
<td><strong>X2MZ</strong></td>
<td>Sens. of mobility to substrate bias @ Vds=0</td>
<td>cm$^2$/volt$^2$-sec</td>
<td>ζ</td>
</tr>
<tr>
<td><strong>X2U0</strong></td>
<td>Sens. of transverse-field mobility degradation effect to substrate bias</td>
<td>volt$^{-2}$</td>
<td>ζ</td>
</tr>
</tbody>
</table>
MOSFET model parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2U1</td>
<td>Sens. of velocity saturation effect to substrate bias</td>
<td>$\mu$/volt$^2$</td>
<td>$\varsigma$</td>
</tr>
<tr>
<td>X3E</td>
<td>Sens. of drain-induced barrier lowering effect to drain bias @ $V_{ds} = V_{dd}$</td>
<td>volt$^{-1}$</td>
<td>$\varsigma$</td>
</tr>
<tr>
<td>X3MS</td>
<td>Sens. of mobility to drain bias @ $V_{ds} = V_{dd}$</td>
<td>cm$^2$/volt$^2$·sec</td>
<td>$\varsigma$</td>
</tr>
<tr>
<td>X3U1</td>
<td>Sens. of velocity saturation effect on drain</td>
<td>$\mu$/volt$^2$</td>
<td>$\varsigma$</td>
</tr>
<tr>
<td>XPART</td>
<td>Gate-oxide capacitance charge model flag.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\text{XPART}=0$ selects a 40/60 drain/source charge partition in saturation, while $\text{XPART}=1$ selects a 0/100 drain/source charge partition.

**level 5: process parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>COX</td>
<td>gate oxide capacitance per unit area</td>
<td>F/m$^2$</td>
<td>0.7E-3</td>
</tr>
<tr>
<td>XJ</td>
<td>junction depth</td>
<td>m</td>
<td>0.1E-6</td>
</tr>
<tr>
<td>DW</td>
<td>channel width correction</td>
<td>m</td>
<td>0.0</td>
</tr>
<tr>
<td>DL</td>
<td>channel length correction</td>
<td>m</td>
<td>0.0</td>
</tr>
<tr>
<td>HDIF</td>
<td>length of heavily doped diffusion contact to gate</td>
<td>m</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**level 5: basic intrinsic parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>VTO</td>
<td>long-channel threshold voltage</td>
<td>V</td>
<td>0.5</td>
</tr>
<tr>
<td>GAMMA</td>
<td>body effect parameter</td>
<td>$\sqrt{V}$</td>
<td>1.0</td>
</tr>
<tr>
<td>PHI</td>
<td>bulk Fermi potential (-2)</td>
<td>V</td>
<td>0.7</td>
</tr>
<tr>
<td>KP</td>
<td>transconductance parameter</td>
<td>A/V$^2$</td>
<td>50.0E-6</td>
</tr>
<tr>
<td>E0</td>
<td>mobility reduction coefficient</td>
<td>V/m</td>
<td>1.0E12</td>
</tr>
<tr>
<td>UCRIT</td>
<td>longitudinal critical field</td>
<td>V/m</td>
<td>2.0E6</td>
</tr>
</tbody>
</table>

**level 5: channel length modulation and charge sharing parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAMBDA</td>
<td>depletion length coefficient (channel length modulation)</td>
<td>0.5</td>
</tr>
<tr>
<td>WETA</td>
<td>narrow-channel effect coefficient</td>
<td>0.25</td>
</tr>
<tr>
<td>LETA</td>
<td>short-channel effect coefficient</td>
<td>0.1</td>
</tr>
</tbody>
</table>
# MOSFET model parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>level 5: impact ionization related parameters</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IBA</td>
<td>first impact ionization coefficient</td>
<td>1/m</td>
<td>0.0</td>
</tr>
<tr>
<td>IBB</td>
<td>second impact ionization coefficient</td>
<td>V/m</td>
<td>3.0E8</td>
</tr>
<tr>
<td>IBN</td>
<td>saturation voltage factor for impact ionization</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td><strong>level 5: intrinsic temperature parameters</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCV</td>
<td>threshold voltage temperature coefficient</td>
<td>V/K</td>
<td>1.0E-3</td>
</tr>
<tr>
<td>BEX</td>
<td>mobility temperature exponent</td>
<td></td>
<td>-1.5</td>
</tr>
<tr>
<td>UCEX</td>
<td>longitudinal critical field temperature exponent</td>
<td></td>
<td>0.8</td>
</tr>
<tr>
<td>IBBT</td>
<td>temperature coefficient for IBB</td>
<td>1/K</td>
<td>9.0E-4</td>
</tr>
<tr>
<td><strong>level 5: matching parameters</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AVTO</td>
<td>area related threshold voltage temperature coefficient</td>
<td>V·m</td>
<td>1.0E-6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>see page 179</td>
</tr>
<tr>
<td>AKP</td>
<td>area related gain mismatch parameter</td>
<td>m</td>
<td>1.0E-6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>see page 179</td>
</tr>
<tr>
<td>AGAMMA</td>
<td>area related body effect mismatch parameter</td>
<td>√V·m</td>
<td>1.0E-6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>see page 179</td>
</tr>
<tr>
<td><strong>level 5: resistance parameters</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RBC</td>
<td>bulk contact resistance</td>
<td>ohm</td>
<td>0.0</td>
</tr>
<tr>
<td>RBSH</td>
<td>bulk layer sheet resistance</td>
<td>ohm/square</td>
<td>0.0</td>
</tr>
<tr>
<td>RDC</td>
<td>drain contact resistance</td>
<td>ohm</td>
<td>0.0</td>
</tr>
<tr>
<td>RGC</td>
<td>gate contact resistance</td>
<td>ohm</td>
<td>0.0</td>
</tr>
<tr>
<td>RGSH</td>
<td>gate layer sheet resistance</td>
<td>ohm/square</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>see page 180</td>
</tr>
<tr>
<td>RSC</td>
<td>source contact resistance</td>
<td>ohm</td>
<td>0.0</td>
</tr>
<tr>
<td><strong>level 5: temperature parameters</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TR1</td>
<td>first-order temperature coefficient for drain, source series resistance</td>
<td>°C⁻¹</td>
<td>0.0</td>
</tr>
<tr>
<td>TR2</td>
<td>second-order temperature coefficient for drain, source series resistance</td>
<td>°C⁻²</td>
<td>0.0</td>
</tr>
<tr>
<td>TRB</td>
<td>temperature coefficient for bulk series resistance</td>
<td>°C⁻¹</td>
<td>0.0</td>
</tr>
<tr>
<td>TRG</td>
<td>temperature coefficient for gate series resistance</td>
<td>°C⁻¹</td>
<td>0.0</td>
</tr>
<tr>
<td>XTI</td>
<td>drain, source junction current temperature exponent</td>
<td></td>
<td>0.0</td>
</tr>
</tbody>
</table>
### MOSFET model parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>level 5: optional parameters</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NSUB</td>
<td>channel doping</td>
<td>meter</td>
<td>see page 180</td>
</tr>
<tr>
<td>THETA</td>
<td>mobility reduction coefficient</td>
<td>volt⁻¹</td>
<td>see page 180</td>
</tr>
<tr>
<td>TOX</td>
<td>oxide thickness</td>
<td>meter</td>
<td>see page 180</td>
</tr>
<tr>
<td>UO</td>
<td>low-field mobility</td>
<td>cm²/volt·sec</td>
<td>see page 180</td>
</tr>
<tr>
<td>VFB</td>
<td>flat-band voltage</td>
<td>volt</td>
<td>see page 180</td>
</tr>
<tr>
<td>VMAX</td>
<td>saturation velocity</td>
<td>meter/sec</td>
<td>see page 180</td>
</tr>
<tr>
<td><strong>level 5: setup parameters</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SATLIM</td>
<td>ratio defining the saturation limit (i_f/i_r)</td>
<td></td>
<td>54.6</td>
</tr>
<tr>
<td><strong>level 6</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0</td>
<td>bulk charge effect coefficient NMOS</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>bulk charge effect coefficient PMOS</td>
<td></td>
<td>4.4</td>
</tr>
<tr>
<td>A1</td>
<td>first non-saturation coefficient NMOS</td>
<td>1/V</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>first non-saturation coefficient PMOS</td>
<td>1/V</td>
<td>0.23</td>
</tr>
<tr>
<td>A2</td>
<td>second non-saturation coefficient NMOS</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>second non-saturation coefficient PMOS</td>
<td></td>
<td>0.08</td>
</tr>
<tr>
<td>A_T</td>
<td>saturation velocity temperature coefficient</td>
<td>m/sec</td>
<td>3.3E4</td>
</tr>
<tr>
<td>BULKMOD</td>
<td>bulk charge model selector:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NMOS</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>PMOS</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>CDSC</td>
<td>drain/source and channel coupling capacitance</td>
<td>F/m²</td>
<td>2.4E-4</td>
</tr>
<tr>
<td>CDSCB</td>
<td>body bias sensitivity of CDSC</td>
<td>F/V·m²</td>
<td>0.0</td>
</tr>
<tr>
<td>DL</td>
<td>channel length reduction on one side</td>
<td>m</td>
<td>0.0</td>
</tr>
<tr>
<td>DROUT</td>
<td>channel length dependent coefficient of the DIBL effect on (R_{out})</td>
<td></td>
<td>0.56</td>
</tr>
<tr>
<td>DSUB</td>
<td>subthreshold DIBL coefficient exponent</td>
<td></td>
<td>DROUT</td>
</tr>
<tr>
<td>DVT0</td>
<td>first coefficient of short-channel effect on threshold voltage</td>
<td></td>
<td>2.2</td>
</tr>
<tr>
<td>DVT1</td>
<td>second coefficient of short-channel effect on threshold voltage</td>
<td></td>
<td>0.53</td>
</tr>
<tr>
<td>DVT2</td>
<td>body bias coefficient of short-channel effect on threshold voltage</td>
<td>1/V</td>
<td>-0.032</td>
</tr>
</tbody>
</table>
### MOSFET model parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>DW</td>
<td>channel width reduction on one side</td>
<td>m</td>
<td>0.0</td>
</tr>
<tr>
<td>ETA0</td>
<td>DIBL coefficient in subthreshold region</td>
<td></td>
<td>0.08</td>
</tr>
<tr>
<td>ETAB</td>
<td>body bias coefficient for the subthreshold DIBL coefficient</td>
<td>1/V</td>
<td>-0.07</td>
</tr>
<tr>
<td>K1</td>
<td>first-order body effect coefficient</td>
<td>1/V</td>
<td>see page 181</td>
</tr>
<tr>
<td>K2</td>
<td>second-order body effect coefficient</td>
<td></td>
<td>see page 181</td>
</tr>
<tr>
<td>K3</td>
<td>narrow width effect coefficient</td>
<td></td>
<td>80.0</td>
</tr>
<tr>
<td>K3B</td>
<td>body effect coefficient of K3</td>
<td>1/V</td>
<td>0.0</td>
</tr>
<tr>
<td>KETA</td>
<td>body bias coefficient of the bulk charge effect.</td>
<td>1/V</td>
<td>-0.047</td>
</tr>
<tr>
<td>KT1</td>
<td>temperature coefficient for threshold voltage</td>
<td>V</td>
<td>-0.11</td>
</tr>
<tr>
<td>KT1L</td>
<td>channel length sensitivity of temperature coefficient for threshold voltage.</td>
<td>V-m</td>
<td>0.0</td>
</tr>
<tr>
<td>KT2</td>
<td>body bias coefficient of the threshold voltage temperature effect</td>
<td></td>
<td>0.022</td>
</tr>
<tr>
<td>NFACTOR</td>
<td>subthreshold swing coefficient</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>NGATE</td>
<td>poly gate doping concentration</td>
<td>1/cm³</td>
<td></td>
</tr>
<tr>
<td>NLX</td>
<td>lateral nonuniform doping coefficient</td>
<td>m</td>
<td>1.74E-7</td>
</tr>
<tr>
<td>NPEAK</td>
<td>peak doping concentration near interface</td>
<td>1/cm³</td>
<td>1.7E17</td>
</tr>
<tr>
<td>NSUB</td>
<td>substrate doping concentration</td>
<td>1/cm³</td>
<td>6.0E16</td>
</tr>
<tr>
<td>PCLM</td>
<td>channel length modulation coefficient</td>
<td></td>
<td>1.3</td>
</tr>
<tr>
<td>PDIBL1</td>
<td>first output resistance DIBL effect coefficient</td>
<td></td>
<td>0.39</td>
</tr>
<tr>
<td>PDIBL2</td>
<td>second output resistance DIBL effect coefficient</td>
<td></td>
<td>0.0086</td>
</tr>
<tr>
<td>PSCBE1</td>
<td>first substrate current body effect coefficient</td>
<td>V/m</td>
<td>4.24E8</td>
</tr>
<tr>
<td>PSCBE2</td>
<td>second substrate current body effect coefficient</td>
<td>m/V</td>
<td>1.0E-5</td>
</tr>
<tr>
<td>PVAG</td>
<td>gate dependence of Early voltage</td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td>RDS0</td>
<td>contact resistance</td>
<td>ohms</td>
<td>0.0</td>
</tr>
<tr>
<td>RDSW</td>
<td>parasitic resistance per unit width</td>
<td>ohms/µm</td>
<td>0.0</td>
</tr>
<tr>
<td>SATMOD</td>
<td>saturation model selector:</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>For semi-empirical output:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>resistance model 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For physical output:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>resistance model 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MOSFET model parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBTHMOD</td>
<td>subthreshold model selector:</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>no subthreshold model 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BSIM1 subthreshold model 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BSIM3 subthreshold model 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BSIM3 subthreshold model</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>using log current 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TNOM</td>
<td>temperature at which parameters are extracted.</td>
<td>deg. C</td>
<td>27</td>
</tr>
<tr>
<td>TOX</td>
<td>gate oxide thickness</td>
<td>m</td>
<td>1.5E-8</td>
</tr>
<tr>
<td>UA</td>
<td>first-order mobility degradation coefficient</td>
<td>m/V</td>
<td>2.25E-9</td>
</tr>
<tr>
<td>UA1</td>
<td>temperature coefficient for UA</td>
<td>m/V</td>
<td>4.31E-9</td>
</tr>
<tr>
<td>UB</td>
<td>second-order mobility degradation coefficient</td>
<td>(m/V)^2</td>
<td>5.87E-19</td>
</tr>
<tr>
<td>UB1</td>
<td>temperature coefficient for UB</td>
<td>(m/V)^2</td>
<td>-7.61E-18</td>
</tr>
<tr>
<td>UC</td>
<td>body effect mobility degradation coefficient</td>
<td>1/V</td>
<td>0.0465</td>
</tr>
<tr>
<td>UC1</td>
<td>temperature coefficient for UC</td>
<td>1/V</td>
<td>-0.056</td>
</tr>
<tr>
<td>UTE</td>
<td>mobility temperature exponent</td>
<td></td>
<td>-1.5</td>
</tr>
<tr>
<td>VOFF</td>
<td>offset voltage in subthreshold region</td>
<td>V</td>
<td>-0.11</td>
</tr>
<tr>
<td>VSAT</td>
<td>saturation velocity at Temp=TNOM</td>
<td>cm/sec</td>
<td>8.0E6</td>
</tr>
<tr>
<td>VTH0</td>
<td>threshold voltage at Vbs=0 for large channel length</td>
<td>V</td>
<td>see page 181</td>
</tr>
<tr>
<td>W0</td>
<td>narrow width effect parameter</td>
<td>m</td>
<td>2.5E-6</td>
</tr>
<tr>
<td>XJ</td>
<td>junction depth</td>
<td>m</td>
<td>1.5E-7</td>
</tr>
<tr>
<td>XPART</td>
<td>charge partitioning coefficient:</td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>no charge model &lt; 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>40/60 partition = 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50/50 partition = 0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0/100 partition = 1.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**level 6 advanced**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIT</td>
<td>capacitance due to interface trapped charge</td>
<td>F/m^2</td>
<td>0.0</td>
</tr>
<tr>
<td>EM</td>
<td>critical electrical field in channel</td>
<td>V/m</td>
<td>4.1E7</td>
</tr>
<tr>
<td>ETA</td>
<td>drain voltage reduction coefficient due to LDD</td>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td>GAMMA1</td>
<td>body effect coefficient near the interface</td>
<td>√V</td>
<td>see page 181</td>
</tr>
<tr>
<td>GAMMA2</td>
<td>body effect coefficient in the bulk</td>
<td>√V</td>
<td>see page 181</td>
</tr>
<tr>
<td>LDD</td>
<td>total length of the LDD region</td>
<td>m</td>
<td>0.0</td>
</tr>
<tr>
<td>LITL</td>
<td>characteristic length related to current depth</td>
<td>m</td>
<td>see page 181</td>
</tr>
</tbody>
</table>
### MOSFET model parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHI</td>
<td>surface potential under strong inversion</td>
<td>V</td>
<td>see page 181</td>
</tr>
<tr>
<td>U0</td>
<td>mobility at Temp=TNOM:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NMOS</td>
<td>cm²/V-sec</td>
<td>670.0</td>
</tr>
<tr>
<td></td>
<td>PMOS</td>
<td>cm²/V-sec</td>
<td>250.0</td>
</tr>
<tr>
<td>VBM</td>
<td>maximum applied body bias</td>
<td>V</td>
<td>-5.0</td>
</tr>
<tr>
<td>VBX</td>
<td>vbs at which the depletion width equals XT</td>
<td>V</td>
<td>see page 181</td>
</tr>
<tr>
<td>VFB</td>
<td>flat-band voltage</td>
<td>V</td>
<td>see page 181</td>
</tr>
<tr>
<td>VGHIGH</td>
<td>voltage shift of the higher bound of the transition</td>
<td>V</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td>region</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VGLOW</td>
<td>voltage shift of the lower bound of the transition</td>
<td>V</td>
<td>-0.12</td>
</tr>
<tr>
<td></td>
<td>region</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XT</td>
<td>doping depth</td>
<td>m</td>
<td>1.55E-7</td>
</tr>
</tbody>
</table>

#### level 7: control parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAPMOD</td>
<td>flag for the short-channel capacitance model</td>
<td>none</td>
<td>2</td>
</tr>
<tr>
<td>MOBMOD</td>
<td>mobility model selector</td>
<td>none</td>
<td>1</td>
</tr>
<tr>
<td>NOIMOD</td>
<td>flag for noise model</td>
<td>none</td>
<td>1</td>
</tr>
<tr>
<td>NQSMOD</td>
<td>flag for NQS model</td>
<td>none</td>
<td>0</td>
</tr>
<tr>
<td>PARAMCHK</td>
<td>flag for model parameter checking</td>
<td>none</td>
<td>0</td>
</tr>
</tbody>
</table>

#### level 7: AC and capacitance parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>CF</td>
<td>fringing field capacitance</td>
<td>F/m</td>
<td>see page 182</td>
</tr>
<tr>
<td>CKAPPA</td>
<td>coefficient for lightly doped region overlap capacitance fringing field capacitance</td>
<td>F/m</td>
<td>0.6</td>
</tr>
<tr>
<td>CLC</td>
<td>constant term for the short-channel model</td>
<td>m</td>
<td>0.1E-6</td>
</tr>
<tr>
<td>CLE</td>
<td>exponential term for the short-channel model</td>
<td>none</td>
<td>0.6</td>
</tr>
<tr>
<td>CGBO</td>
<td>gate-bulk overlap capacitance per unit channel length</td>
<td>F/m</td>
<td>0.0</td>
</tr>
<tr>
<td>CGDL</td>
<td>light-doped drain-gate region overlap capacitance</td>
<td>F/m</td>
<td>0.0</td>
</tr>
<tr>
<td>CGDO</td>
<td>non-LDD region drain-gate overlap capacitance</td>
<td>F/m</td>
<td>see page 183</td>
</tr>
<tr>
<td></td>
<td>per channel length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CGSL</td>
<td>light-doped source-gate region overlap capacitance</td>
<td>F/m</td>
<td>0.0</td>
</tr>
<tr>
<td>CGSO</td>
<td>non-LDD region source-gate overlap capacitance</td>
<td>F/m</td>
<td>see page 183</td>
</tr>
<tr>
<td></td>
<td>per channel length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CJ</td>
<td>bottom junction capacitance per unit area</td>
<td>F/m²</td>
<td>5.0E-4</td>
</tr>
</tbody>
</table>
### MOSFET model parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>CJSW</td>
<td>source/drain side junction capacitance per unit periphery</td>
<td>F/m</td>
<td>5.0E-10</td>
</tr>
<tr>
<td>CJSWG</td>
<td>source/drain gate sidewall junction capacitance per unit width</td>
<td>F/m</td>
<td>CJSW</td>
</tr>
<tr>
<td>DLC</td>
<td>length offset fitting parameter from C-V</td>
<td>m</td>
<td>LINT</td>
</tr>
<tr>
<td>DWC</td>
<td>width offset fitting parameter from C-V</td>
<td>m</td>
<td>WINT</td>
</tr>
<tr>
<td>MJ</td>
<td>bottom junction capacitance grading coefficient</td>
<td>none</td>
<td>0.5</td>
</tr>
<tr>
<td>MJSW</td>
<td>source/drain side junction capacitance grading coefficient</td>
<td>none</td>
<td>0.33</td>
</tr>
<tr>
<td>MJSWG</td>
<td>source/drain gate sidewall junction capacitance grading coefficient</td>
<td>none</td>
<td>MJSW</td>
</tr>
<tr>
<td>PB</td>
<td>bottom built-in potential</td>
<td>V</td>
<td>1.0</td>
</tr>
<tr>
<td>PBSW</td>
<td>source/drain side junction built-in potential</td>
<td>V</td>
<td>PBSW</td>
</tr>
<tr>
<td>PBSWG</td>
<td>source/drain gate sidewall junction built-in potential</td>
<td>V</td>
<td>PBSW</td>
</tr>
<tr>
<td>VFBCV</td>
<td>flat-band voltage parameter</td>
<td>V</td>
<td>-1.0</td>
</tr>
<tr>
<td>XPART</td>
<td>charge partitioning rate flag</td>
<td>none</td>
<td>0.0</td>
</tr>
</tbody>
</table>

#### level 7: bin description parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>BINUNIT</td>
<td>bin unit scale selector</td>
<td>none</td>
<td>1.0</td>
</tr>
<tr>
<td>LMAX</td>
<td>maximum channel length</td>
<td>m</td>
<td>1.0</td>
</tr>
<tr>
<td>LMIN</td>
<td>minimum channel length</td>
<td>m</td>
<td>0.0</td>
</tr>
<tr>
<td>WMAX</td>
<td>maximum channel width</td>
<td>m</td>
<td>1.0</td>
</tr>
<tr>
<td>WMIN</td>
<td>minimum channel width</td>
<td>m</td>
<td>0.0</td>
</tr>
</tbody>
</table>

#### level 7: DC parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>bulk charge effect coefficient for channel length</td>
<td>none</td>
<td>1.0</td>
</tr>
<tr>
<td>A1</td>
<td>first non-saturation effect parameter</td>
<td>1/V</td>
<td>0.0</td>
</tr>
<tr>
<td>A2</td>
<td>second non-saturation factor</td>
<td>none</td>
<td>1.0</td>
</tr>
<tr>
<td>AGS</td>
<td>gate-bias coefficient of Abulk</td>
<td>1/V</td>
<td>0.0</td>
</tr>
<tr>
<td>ALPHA0</td>
<td>first parameter of impact-ionization current</td>
<td>m/V</td>
<td>0.0</td>
</tr>
<tr>
<td>B0</td>
<td>bulk charge effect coefficient for channel width</td>
<td>m</td>
<td>0.0</td>
</tr>
<tr>
<td>B1</td>
<td>bulk charge effect width offset</td>
<td>m</td>
<td>0.0</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Unit</td>
<td>Default</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------------</td>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>BETA0</td>
<td>second parameter of impact-ionization current</td>
<td>V</td>
<td>30.0</td>
</tr>
<tr>
<td>CDSC</td>
<td>drain/source to channel coupling capacitance</td>
<td>F/m²</td>
<td>2.4E-4</td>
</tr>
<tr>
<td>CDSCB</td>
<td>body-bias sensitivity of CDSC</td>
<td>F/Vm²</td>
<td>0.0</td>
</tr>
<tr>
<td>CDSCD</td>
<td>drain-bias sensitivity of CDSC</td>
<td>F/Vm²</td>
<td>0.0</td>
</tr>
<tr>
<td>CIT</td>
<td>interface trap capacitance</td>
<td>F/m²</td>
<td>0.0</td>
</tr>
<tr>
<td>DELTA</td>
<td>effective Vds parameter</td>
<td>V</td>
<td>0.01</td>
</tr>
<tr>
<td>DROUT</td>
<td>L-dependence coefficient of the DIBL correction parameter in Rout</td>
<td>none</td>
<td>0.56</td>
</tr>
<tr>
<td>DSUB</td>
<td>DIBL coefficient exponent in subthreshold region</td>
<td>none</td>
<td>DROUT</td>
</tr>
<tr>
<td>DVT0</td>
<td>first coefficient of short-channel effect on threshold voltage</td>
<td>none</td>
<td>2.2</td>
</tr>
<tr>
<td>DVT0W</td>
<td>first coefficient of narrow-width effect on threshold voltage for small-channel length</td>
<td>l/m</td>
<td>0.0</td>
</tr>
<tr>
<td>DVT1</td>
<td>second coefficient of short-channel effect on threshold voltage</td>
<td>none</td>
<td>0.53</td>
</tr>
<tr>
<td>DVT2</td>
<td>body-bias coefficient of short-channel effect on threshold voltage</td>
<td>l/V</td>
<td>-0.032</td>
</tr>
<tr>
<td>DVTW1</td>
<td>second coefficient of narrow-width effect on threshold voltage for small channel length</td>
<td>l/m</td>
<td>5.3E6</td>
</tr>
<tr>
<td>DVTW2</td>
<td>body-bias coefficient of narrow-width effect for small channel length</td>
<td>l/V</td>
<td>-0.032</td>
</tr>
<tr>
<td>DWB</td>
<td>coefficient of substrate body bias dependence of Weff</td>
<td>m/V⁹</td>
<td>0.0</td>
</tr>
<tr>
<td>DWG</td>
<td>coefficient of gate dependence of Weff</td>
<td>m/V</td>
<td>0.0</td>
</tr>
<tr>
<td>ETA0</td>
<td>DIBL coefficient in subthreshold region</td>
<td>none</td>
<td>0.08</td>
</tr>
<tr>
<td>ETAB</td>
<td>body-bias coefficient for the subthreshold DIBL effect</td>
<td>l/V</td>
<td>-0.07</td>
</tr>
<tr>
<td>JS</td>
<td>source-drain junction saturation current per unit area</td>
<td>A/m²</td>
<td>1.0E-4</td>
</tr>
<tr>
<td>JSW</td>
<td>sidewall saturation current per unit length</td>
<td>A/m</td>
<td>0.0</td>
</tr>
<tr>
<td>K1</td>
<td>first-order body effect coefficient</td>
<td>V⁹/²</td>
<td>0.5</td>
</tr>
<tr>
<td>K2</td>
<td>second-order body effect coefficient</td>
<td>none</td>
<td>0.0</td>
</tr>
<tr>
<td>K3</td>
<td>narrow width coefficient</td>
<td>none</td>
<td>80.0</td>
</tr>
</tbody>
</table>
## MOSFET model parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>K3B</td>
<td>body effect coefficient of K3</td>
<td>1/V</td>
<td>0.0</td>
</tr>
<tr>
<td>KETA</td>
<td>body-bias coefficient of bulk charge effect</td>
<td>1/V</td>
<td>-0.047</td>
</tr>
<tr>
<td>LINT</td>
<td>length offset fitting parameter from I-V without bias</td>
<td>m</td>
<td>0.0</td>
</tr>
<tr>
<td>NFACTOR</td>
<td>subthreshold swing factor</td>
<td>none</td>
<td>1.0</td>
</tr>
<tr>
<td>NGATE</td>
<td>poly gate doping concentration</td>
<td>cm⁻³</td>
<td>0.0</td>
</tr>
<tr>
<td>NLX</td>
<td>lateral non-uniform doping parameter</td>
<td>m</td>
<td>1.74E-7</td>
</tr>
<tr>
<td>PCLM</td>
<td>channel length modulation parameter</td>
<td>none</td>
<td>1.3</td>
</tr>
<tr>
<td>PDIBLC1</td>
<td>first output resistance DIBL effect correction parameter</td>
<td>none</td>
<td>0.39</td>
</tr>
<tr>
<td>PDIBLC2</td>
<td>second output resistance DIBL effect correction parameter</td>
<td>none</td>
<td>0.0086</td>
</tr>
<tr>
<td>PDIBLCB</td>
<td>body effect coefficient of DIBL correction parameter</td>
<td>1/V</td>
<td>0.0</td>
</tr>
<tr>
<td>PRWB</td>
<td>body effect coefficient of RDSW</td>
<td>1/V¹/²</td>
<td>0.0</td>
</tr>
<tr>
<td>PRWG</td>
<td>gate-bias effect coefficient of RDSW</td>
<td>1/V</td>
<td>0.0</td>
</tr>
<tr>
<td>PSCBE1</td>
<td>first substrate current body effect parameter</td>
<td>V/m</td>
<td>4.24E8</td>
</tr>
<tr>
<td>PSCBE2</td>
<td>second substrate current body effect parameter</td>
<td>V/m</td>
<td>1.0E-5</td>
</tr>
<tr>
<td>PVAG</td>
<td>gate dependence of Early voltage</td>
<td>none</td>
<td>0.0</td>
</tr>
<tr>
<td>RDSW</td>
<td>parasitic resistance per unit width</td>
<td>Ω⋅μmWR</td>
<td>0.0</td>
</tr>
<tr>
<td>RSH</td>
<td>source-drain sheet resistance</td>
<td>Ω/square</td>
<td>0.0</td>
</tr>
<tr>
<td>U0</td>
<td>mobility at Temp=TNOM</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NMOS</td>
<td>670.0</td>
<td>cm²/(V⋅sec)</td>
</tr>
<tr>
<td></td>
<td>PMOS</td>
<td>250.0</td>
<td></td>
</tr>
<tr>
<td>UA</td>
<td>first-order mobility degradation coefficient</td>
<td>m/V</td>
<td>2.25E-9</td>
</tr>
<tr>
<td>UB</td>
<td>second-order mobility degradation coefficient</td>
<td>(m/V)²</td>
<td>5.87E-19</td>
</tr>
<tr>
<td>UC</td>
<td>body effect of mobility degradation coefficient</td>
<td>m/V²</td>
<td>-4.65E-11 when MOBMOD=1 or 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1/V</td>
<td>-0.046 when MOBMOD=3</td>
</tr>
<tr>
<td>VBM</td>
<td>maximum applied body-bias in threshold voltage calculation</td>
<td>V</td>
<td>-3.0</td>
</tr>
<tr>
<td>VOFF</td>
<td>offset voltage in the subthreshold region at large W and L</td>
<td>V</td>
<td>-0.08</td>
</tr>
</tbody>
</table>
## MOSFET model parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSAT</td>
<td>saturation velocity at Temp=TNOM</td>
<td>m/sec</td>
<td>8.0E-4</td>
</tr>
<tr>
<td>VTH0</td>
<td>threshold voltage@Vbs=0 for large L</td>
<td>V</td>
<td>0.7 (NMOS) &lt;br&gt;-0.7 (PMOS)</td>
</tr>
<tr>
<td>W0</td>
<td>narrow-width parameter</td>
<td>m</td>
<td>2.5E-6</td>
</tr>
<tr>
<td>WINT</td>
<td>width-offset fitting parameter from I-V without bias</td>
<td>m</td>
<td>0.0</td>
</tr>
<tr>
<td>WR</td>
<td>width-offset from Weff for Rds calculation</td>
<td>none</td>
<td>1.0</td>
</tr>
</tbody>
</table>

### Level 7: flicker noise parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF</td>
<td>frequency exponent</td>
<td>none</td>
<td>1.0</td>
</tr>
<tr>
<td>EF</td>
<td>flicker exponent</td>
<td>none</td>
<td>1.0</td>
</tr>
<tr>
<td>EM</td>
<td>saturation field</td>
<td>V/m</td>
<td>4.1E7</td>
</tr>
<tr>
<td>KF</td>
<td>flicker noise parameter</td>
<td>none</td>
<td>0.0</td>
</tr>
<tr>
<td>NOIA</td>
<td>noise parameter A</td>
<td>none</td>
<td>1.0E20 (NMOS) &lt;br&gt;9.9E18 (PMOS)</td>
</tr>
<tr>
<td>NOIB</td>
<td>noise parameter B</td>
<td>none</td>
<td>5.0E4 (NMOS) &lt;br&gt;2.4E3 (PMOS)</td>
</tr>
<tr>
<td>NOIC</td>
<td>noise parameter C</td>
<td>none</td>
<td>-1.4E-12 (NMOS) &lt;br&gt;1.4E-12 (PMOS)</td>
</tr>
</tbody>
</table>

### Level 7: NQS parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELM</td>
<td>Elmore constant of the channel</td>
<td>none</td>
<td>5.0</td>
</tr>
</tbody>
</table>

### Level 7: process parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAMMA1</td>
<td>body effect coefficient near the surface</td>
<td>V^{1/2}</td>
<td>see page 182</td>
</tr>
<tr>
<td>GAMMA2</td>
<td>body effect coefficient in the bulk</td>
<td>V^{1/2}</td>
<td>see page 182</td>
</tr>
<tr>
<td>NCH</td>
<td>channel doping concentration</td>
<td>1/cm³</td>
<td>1.7E17</td>
</tr>
<tr>
<td>NSUB</td>
<td>substrate doping concentration</td>
<td>1/cm³</td>
<td>6.0E16</td>
</tr>
<tr>
<td>TOX</td>
<td>gate-oxide thickness</td>
<td>m</td>
<td>1.5E-8</td>
</tr>
<tr>
<td>VBX</td>
<td>Vbs at which the depletion region = XT</td>
<td>V</td>
<td>see page 182</td>
</tr>
<tr>
<td>XJ</td>
<td>junction depth</td>
<td>m</td>
<td>1.5E-7</td>
</tr>
<tr>
<td>XT</td>
<td>doping depth</td>
<td>m</td>
<td>1.55E-7</td>
</tr>
</tbody>
</table>

### Level 7: temperature parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT</td>
<td>temperature coefficient for saturation velocity</td>
<td>m/sec</td>
<td>3.3E4</td>
</tr>
</tbody>
</table>
### MOSFET model parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>KT1</td>
<td>temperature coefficient for threshold voltage</td>
<td>V</td>
<td>-0.11</td>
</tr>
<tr>
<td>KT1L</td>
<td>channel length dependence of the temperature coefficient for threshold voltage</td>
<td>V*m</td>
<td>0.0</td>
</tr>
<tr>
<td>KT2</td>
<td>body-bias coefficient of threshold voltage temperature effect</td>
<td>none</td>
<td>0.022</td>
</tr>
<tr>
<td>NJ</td>
<td>emission coefficient of junction</td>
<td>none</td>
<td>1.0</td>
</tr>
<tr>
<td>PRT</td>
<td>temperature coefficient for $R_{DSW}$</td>
<td>Ω⋅µm</td>
<td>0.0</td>
</tr>
<tr>
<td>TNOM</td>
<td>temperature at which parameters are extracted</td>
<td>°C</td>
<td>27.0</td>
</tr>
<tr>
<td>UA1</td>
<td>temperature coefficient for $U_A$</td>
<td>m/V</td>
<td>4.31E-9</td>
</tr>
<tr>
<td>UB1</td>
<td>temperature coefficient for $U_B$</td>
<td>(m/V)</td>
<td>-7.61E-18</td>
</tr>
<tr>
<td>UC1</td>
<td>temperature coefficient for $U_C$</td>
<td>m/V²</td>
<td>-5.6E-11 when $MOBMOD=1$ or 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-0.056 when $MOBMOD=3$</td>
</tr>
<tr>
<td>UTE</td>
<td>mobility temperature exponent</td>
<td>none</td>
<td>-1.5</td>
</tr>
<tr>
<td>XTI</td>
<td>junction current temperature exponent coefficient</td>
<td>none</td>
<td>3.0</td>
</tr>
</tbody>
</table>

#### level 7: W and L parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Unit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>LL</td>
<td>coefficient of length dependence for length offset</td>
<td>m^LLN</td>
<td>0.0</td>
</tr>
<tr>
<td>LLN</td>
<td>power of length dependence for length offset</td>
<td>none</td>
<td>1.0</td>
</tr>
<tr>
<td>LW</td>
<td>coefficient of width dependence for length offset</td>
<td>m^LWN</td>
<td>0.0</td>
</tr>
<tr>
<td>LWL</td>
<td>coefficient of length and width cross term for length offset</td>
<td>m^LWN+LLN</td>
<td>0.0</td>
</tr>
<tr>
<td>LWN</td>
<td>power of width dependence for length offset</td>
<td>none</td>
<td>1.0</td>
</tr>
<tr>
<td>WL</td>
<td>coefficient of length dependence for width offset</td>
<td>m^WLN</td>
<td>0.0</td>
</tr>
<tr>
<td>WLN</td>
<td>power of length dependence of width offset</td>
<td>none</td>
<td>1.0</td>
</tr>
<tr>
<td>WW</td>
<td>coefficient of width dependence for width offset</td>
<td>m^WWN</td>
<td>0.0</td>
</tr>
<tr>
<td>WWL</td>
<td>coefficient of length and width cross term for width offset</td>
<td>m^WWN+WLN</td>
<td>0.0</td>
</tr>
<tr>
<td>WWN</td>
<td>power of width dependence of width offset</td>
<td>none</td>
<td>1.0</td>
</tr>
</tbody>
</table>

* See .MODEL (model definition).
** A ζ in the Default column indicates that the parameter may have corresponding parameters exhibiting length and width dependence. See Model level 4.
† For information on $T_{MEASURED}$, $T_{ABS}$, $T_{REL\_GLOBAL}$, and $T_{REL\_LOCAL}$, see .MODEL (model definition).
MOSFET Equations

These equations describe an N-channel MOSFET. For P-channel devices, reverse the signs of all voltages and currents.

In the following equations:

- $V_{bs} = \text{intrinsic substrate-intrinsic source voltage}$
- $V_{bd} = \text{intrinsic substrate-intrinsic drain voltage}$
- $V_{ds} = \text{intrinsic drain-intrinsic source voltage}$
- $V_{dsat} = \text{saturation voltage}$
- $V_{gs} = \text{intrinsic gate-intrinsic source voltage}$
- $V_{gd} = \text{intrinsic gate-intrinsic drain voltage}$
- $V_t = \frac{k \cdot T}{q} \text{ (thermal voltage)}$
- $V_{th} = \text{threshold voltage}$
- $C_{ox} = \text{the gate oxide capacitance per unit area.}$
- $f = \text{noise frequency}$
- $k = \text{Boltzmann’s constant}$
- $q = \text{electron charge}$
- $L_{eff} = \text{effective channel length}$
- $W_{eff} = \text{effective channel width}$
- $T = \text{analysis temperature (°K)}$
- $T_{nom} = \text{nominal temperature (set using TNOM option)}$

Other variables are from [MOSFET model parameters](#).

Positive current is current flowing into a terminal (for example, positive drain current flows from the drain through the channel to the source).
MOSFET equations for DC current

<table>
<thead>
<tr>
<th>all levels</th>
</tr>
</thead>
</table>

Ig = gate current = 0

Ib = bulk current = lbs+Ibd

where

Ibs = bulk-source leakage current = Iss · (e^{\frac{V_{bs}}{N \cdot V_t}} - 1)

Ibd = bulk-drain leakage current = Ids · (e^{\frac{V_{bd}}{N \cdot V_t}} - 1)

where

if

JS = 0, or AS = 0, or AD = 0

then

Iss = IS

Ids = IS

else

Iss = AS · JS + PS · JSSW

Ids = AD · JS + PD · JSSW

Id = drain current = Idrain-Ibd

Is = source current = -Idrain-Ibs

level 1: Idrain

Normal mode: Vds > 0

Case 1

for cutoff region: Vgs-V_{to} < 0

then: Idrain = 0

Case 2

for linear region: Vds < Vgs-V_{to}

then: Idrain = \frac{W}{L} · \frac{K_P}{2} · (1+\lambda \cdot V_{ds}) · V_{ds} · (2 · (Vgs-V_{to}) - V_{ds})

Case 3

for saturation region: 0 < Vgs-V_{to} < Vds

then: Idrain = \frac{W}{L} · \frac{K_P}{2} · (1+\lambda \cdot V_{ds}) · (Vgs-V_{to})^2

where

V_{to} = V_{TO} + \gamma \cdot \left( (\phi - V_{bs})^{1/2} - \phi^{1/2} \right)

Inverted mode: Vds < 0

Switch the source and drain in the normal mode equations above.

Levels 2 and 3: Idrain

MOSFET equations for capacitance

All capacitances are between terminals of the intrinsic MOSFET, in other words, to the inside of the ohmic drain and source resistances. For levels 1, 2, and 3, the capacitance model has been changed to conserve charge.

**levels 1, 2, and 3**

\[ C_{bs} = \text{bulk-source capacitance} = \text{area cap.} + \text{sidewall cap.} + \text{transit time cap.} \]

\[ C_{bd} = \text{bulk-drain capacitance} = \text{area cap.} + \text{sidewall cap.} + \text{transit time cap.} \]

where

if

\[ C_{bs} = 0 \ AND \ C_{bd} = 0 \]

then

\[ C_{bs} = A \cdot C_{J} \cdot C_{bsj} + P \cdot C_{JSW} \cdot C_{bss} + T \cdot G_{bs} \]

\[ C_{bd} = A \cdot C_{J} \cdot C_{bdj} + P \cdot C_{JSW} \cdot C_{bds} + T \cdot G_{ds} \]

else

\[ C_{bs} = C_{bs} \cdot C_{bsj} + P \cdot C_{JSW} \cdot C_{bss} + T \cdot G_{bs} \]

\[ C_{bd} = C_{bd} \cdot C_{bdj} + P \cdot C_{JSW} \cdot C_{bds} + T \cdot G_{ds} \]

where

\[ G_{bs} = \text{DC bulk-source conductance} = \frac{dI_{bs}}{dV_{bs}} \]

\[ G_{bd} = \text{DC bulk-drain conductance} = \frac{dI_{bd}}{dV_{bd}} \]

if

\[ V_{bs} \leq F \cdot P \]

then

\[ C_{bsj} = (1 - V_{bs}/P)^{\cdot}^{M_{J}} \]

\[ C_{bss} = (1 - V_{bs}/P_{SW})^{\cdot}^{M_{JSW}} \]

if

\[ V_{bs} > F \cdot P \]

then

\[ C_{bsj} = (1 - F \cdot (1 + M_{J}) \cdot M_{J} \cdot V_{bs}/P)^{\cdot} \]

\[ C_{bss} = (1 - F \cdot (1 + M_{JSW}) + M_{JSW} \cdot V_{bs}/P_{SW})^{\cdot} \]

if

\[ V_{bd} \leq F \cdot P \]

then

\[ C_{bdj} = (1 - V_{bd}/P)^{\cdot}^{M_{J}} \]

\[ C_{bds} = (1 - V_{bd}/P_{SW})^{\cdot}^{M_{JSW}} \]

if

\[ V_{bd} > F \cdot P \]

then

\[ C_{bdj} = (1 - F \cdot (1 + M_{J}) \cdot M_{J} \cdot V_{bd}/P)^{\cdot} \]

\[ C_{bds} = (1 - F \cdot (1 + M_{JSW}) + M_{JSW} \cdot V_{bd}/P_{SW})^{\cdot} \]
Cgs = gate-source overlap capacitance = \( C_{GSO} \cdot W \)
Cgd = gate-drain overlap capacitance = \( C_{GDO} \cdot W \)
Cgb = gate-bulk overlap capacitance = \( C_{GBO} \cdot L \)

**Levels 4 and 6**


**MOSFET equations for temperature effects**

The ohmic (parasitic) resistances have no temperature dependence.

\[
\begin{align*}
I_S(T) &= I_S \cdot e^{ \left( E_g(T_{nom}) \cdot T / T_{nom} - E_g(T) / V_t \right)} \\
J_S(T) &= J_S \cdot e^{ \left( E_g(T_{nom}) \cdot T / T_{nom} - E_g(T) / V_t \right)} \\
J_{SSW}(T) &= J_{SSW} \cdot e^{ \left( E_g(T_{nom}) \cdot T / T_{nom} - E_g(T) / V_t \right)} \\
P_B(T) &= P_B \cdot T / T_{nom} - 3 \cdot V_t \cdot \ln \left( T / T_{nom} \right) - E_g(T_{nom}) \cdot T / T_{nom} + E_g(T) \\
P_{BSW}(T) &= P_{BSW} \cdot T / T_{nom} - 3 \cdot V_t \cdot \ln \left( T / T_{nom} \right) - E_g(T_{nom}) \cdot T / T_{nom} + E_g(T) \\
\Phi(T) &= \Phi \cdot T / T_{nom} - 3 \cdot V_t \cdot \ln \left( T / T_{nom} \right) - E_g(T_{nom}) \cdot T / T_{nom} + E_g(T) \\
\text{where} \quad E_g(T) &= \text{silicon bandgap energy} = 1.16 - .000702 \cdot T^2 / (T + 1108) \\
C_{BD}(T) &= C_{BD} \cdot (1 + M_{J} \cdot \left( .0004 \cdot (T - T_{nom}) + (1 - P_B(T) / P_B) \right)) \\
C_{BS}(T) &= C_{BS} \cdot (1 + M_{J} \cdot \left( .0004 \cdot (T - T_{nom}) + (1 - P_B(T) / P_B) \right)) \\
C_{J}(T) &= C_{J} \cdot (1 + M_{J} \cdot \left( .0004 \cdot (T - T_{nom}) + (1 - P_B(T) / P_B) \right)) \\
C_{JSW}(T) &= C_{JSW} \cdot (1 + M_{JSW} \cdot \left( .0004 \cdot (T - T_{nom}) + (1 - P_B(T) / P_B) \right)) \\
K_{P}(T) &= K_{P} \cdot (T / T_{nom})^{-3/2} \\
U_O(T) &= U_O \cdot (T / T_{nom})^{-3/2} \\
M_{US}(T) &= M_{US} \cdot (T / T_{nom})^{-3/2} \\
M_{UZ}(T) &= M_{UZ} \cdot (T / T_{nom})^{-3/2} \\
X_{3MS}(T) &= X_{3MS} \cdot (T / T_{nom})^{-3/2}
\end{align*}
\]
MOSFET equations for noise

Noise is calculated assuming a 1.0-hertz bandwidth, using the following spectral power densities (per unit bandwidth).

The model parameter \( NLEV \) is used to select the form of shot and flicker noise, and \( GDSNOI \) is the channel shot noise coefficient model parameter. When \( NLEV < 3 \), the original SPICE2 shot noise equation is used in both the linear and saturation regions, but the use of this equation may produce inaccurate results in the linear region. When \( NLEV = 3 \), a different equation is used that is valid in both linear and saturation regions.

The model parameters \( AF \) and \( KF \) are used in the small-signal AC noise analysis to determine the equivalent MOSFET flicker noise.

For more information, see reference [5] of References.

**MOSFET channel shot and flicker noise**

\[
I_{chan}^2 = I_{shot}^2 + I_{flick}^2
\]

<table>
<thead>
<tr>
<th>intrinsic MOSFET flicker noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>for ( NLEV = 0 )</td>
</tr>
<tr>
<td>for ( NLEV = 1 )</td>
</tr>
<tr>
<td>for ( NLEV = 2, NLEV = 3 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>intrinsic MOSFET shot noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>for ( NLEV &lt; 3 )</td>
</tr>
<tr>
<td>for ( NLEV = 3 )</td>
</tr>
</tbody>
</table>

where
- for linear region:
  \( a = 1 - (Vds/V_{dsat}) \)
- for saturation region:
  \( a = 0 \)

<table>
<thead>
<tr>
<th>parasitic resistance thermal noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>( RD )</td>
</tr>
<tr>
<td>( RG )</td>
</tr>
<tr>
<td>( RS )</td>
</tr>
<tr>
<td>( RB )</td>
</tr>
</tbody>
</table>
References

For a more complete description of the MOSFET models, refer to:


For more information on References [2] and [4], contact:

Software Distribution Office
EECS/ERL Industrial Liaison Program
205 Cory Hall #1770
University of California
Berkeley, CA 94720-1770
(510) 643-6687
Bipolar transistor

General form

Q<name> < collector node> <base node> <emitter node>
+ [substrate node] <model name> [area value]

Examples

Q1 14 2 13 PNPNOM
Q13 15 3 0 1 NPNSTRONG 1.5
Q7 VC 5 12 [SUB] LATPNP

Model form

.MODEL <model name> NPN [model parameters]
.MODEL <model name> PNP [model parameters]
.MODEL <model name> LPNP [model parameters]

Arguments and options

[substrate node]

is optional, and if not specified, the default is the ground.

Because the simulator allows alphanumeric names for nodes, and because there is no easy way to distinguish these from the model names, the name (not a number) used for the substrate node needs to be enclosed with square brackets [ ]. Otherwise, nodes would be interpreted as model names. See the third example.

[area value]

is the relative device area and has a default value of 1.

Description

The bipolar transistor is modeled as an intrinsic transistor using ohmic resistances in series with the collector (RC/area), with the base (value varies with current, see Bipolar transistor equations), and with the emitter (RE/area).

Positive current is current flowing into a terminal.
For model parameters with alternate names, such as VAF and VA (the alternate name is shown by using parentheses), either name can be used.

For model types NPN and PNP, the isolation junction capacitance is connected between the intrinsic-collector and substrate nodes. This is the same as in SPICE2, or SPICE3, and works well for vertical IC transistor structures. For lateral IC transistor structures there is a third model, LPNP, where the isolation junction capacitance is connected between the intrinsic-base and substrate nodes.

## Capture parts

The following table lists the set of bipolar transistor breakout parts designed for customizing model parameters for simulation. These are useful for setting up Monte Carlo and worst-case analyses with device and/or lot tolerances specified for individual model parameters.

<table>
<thead>
<tr>
<th>Part name</th>
<th>Model type</th>
<th>Property</th>
<th>Property description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QBREAKL</td>
<td>LPNP</td>
<td>AREA</td>
<td>area scaling factor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MODEL</td>
<td>LNP model name</td>
</tr>
<tr>
<td>QBREAKN</td>
<td>NPN</td>
<td>AREA</td>
<td>area scaling factor</td>
</tr>
<tr>
<td>QBREAKN3</td>
<td></td>
<td>MODEL</td>
<td>NPN model name</td>
</tr>
<tr>
<td>QBREAKN4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QBREAKP</td>
<td>PNP</td>
<td>AREA</td>
<td>area scaling factor</td>
</tr>
<tr>
<td>QBREAKP3</td>
<td></td>
<td>MODEL</td>
<td>PNP model name</td>
</tr>
<tr>
<td>QBREAKP4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Setting operating temperature

Operating temperature can be set to be different from the global circuit temperature by defining one of the model parameters: T_ABS, T_REL_GLOBAL, or T_REL_LOCAL. Additionally, model parameters can be assigned unique measurement temperatures using the T_MEASURED model parameter. See [Bipolar transistor model parameters](#) for more information.
<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF</td>
<td>flicker noise exponent</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>BF</td>
<td>ideal maximum forward beta</td>
<td></td>
<td>100.0</td>
</tr>
<tr>
<td>BR</td>
<td>ideal maximum reverse beta</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>CJC</td>
<td>base-collector zero-bias p-n capacitance</td>
<td>farad</td>
<td>0.0</td>
</tr>
<tr>
<td>CJE</td>
<td>base-emitter zero-bias p-n capacitance</td>
<td>farad</td>
<td>0.0</td>
</tr>
<tr>
<td>CJS (CCS)</td>
<td>substrate zero-bias p-n capacitance</td>
<td>farad</td>
<td>0.0</td>
</tr>
<tr>
<td>CN</td>
<td>quasi-saturation temperature coefficient for hole mobility</td>
<td></td>
<td>2.42 NPN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.20 PNP</td>
</tr>
<tr>
<td>D</td>
<td>quasi-saturation temperature coefficient for scattering-limited hole mobility</td>
<td></td>
<td>0.87 NPN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.52 PNP</td>
</tr>
<tr>
<td>EG</td>
<td>bandgap voltage (barrier height)</td>
<td>eV</td>
<td>1.11</td>
</tr>
<tr>
<td>FC</td>
<td>forward-bias depletion capacitor coefficient</td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>GAMMA</td>
<td>epitaxial region doping factor</td>
<td></td>
<td>1E-11</td>
</tr>
<tr>
<td>IKF (IK)</td>
<td>corner for forward-beta high-current roll-off</td>
<td>amp</td>
<td>infinite</td>
</tr>
<tr>
<td>IKR</td>
<td>corner for reverse-beta high-current roll-off</td>
<td>amp</td>
<td>infinite</td>
</tr>
<tr>
<td>IRB</td>
<td>current at which Rb falls halfway to</td>
<td>amp</td>
<td>infinite</td>
</tr>
<tr>
<td>IS</td>
<td>transport saturation current</td>
<td>amp</td>
<td>1E-16</td>
</tr>
<tr>
<td>ISC (C4) †</td>
<td>base-collector leakage saturation current</td>
<td>amp</td>
<td>0.0</td>
</tr>
<tr>
<td>ISE (C2) †</td>
<td>base-emitter leakage saturation current</td>
<td>amp</td>
<td>0.0</td>
</tr>
<tr>
<td>ISS</td>
<td>substrate p-n saturation current</td>
<td>amp</td>
<td>0.0</td>
</tr>
<tr>
<td>ITF</td>
<td>transit time dependency on Ic</td>
<td>amp</td>
<td>0.0</td>
</tr>
<tr>
<td>KF</td>
<td>flicker noise coefficient</td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td>MJC (MC)</td>
<td>base-collector p-n grading factor</td>
<td></td>
<td>0.33</td>
</tr>
<tr>
<td>MJE (ME)</td>
<td>base-emitter p-n grading factor</td>
<td></td>
<td>0.33</td>
</tr>
<tr>
<td>MJS (MS)</td>
<td>substrate p-n grading factor</td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td>NC</td>
<td>base-collector leakage emission coefficient</td>
<td></td>
<td>2.0</td>
</tr>
<tr>
<td>NE</td>
<td>base-emitter leakage emission coefficient</td>
<td></td>
<td>1.5</td>
</tr>
<tr>
<td>NF</td>
<td>forward current emission coefficient</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>NK</td>
<td>high-current roll-off coefficient</td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>NR</td>
<td>reverse current emission coefficient</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>Model parameters</td>
<td>Description</td>
<td>Units</td>
<td>Default</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>NS</td>
<td>substrate p-n emission coefficient</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>PTF</td>
<td>excess phase @ 1/(2π·TF)Hz</td>
<td>degree</td>
<td>0.0</td>
</tr>
<tr>
<td>QCO</td>
<td>epitaxial region charge factor</td>
<td>coulomb</td>
<td>0.0</td>
</tr>
<tr>
<td>QUASIMOD</td>
<td>quasi-saturation model flag for temperature dependence</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>RB</td>
<td>zero-bias (maximum) base resistance</td>
<td>ohm</td>
<td>0.0</td>
</tr>
<tr>
<td>RBM</td>
<td>minimum base resistance</td>
<td>ohm</td>
<td>RB</td>
</tr>
<tr>
<td>RC</td>
<td>collector ohmic resistance</td>
<td>ohm</td>
<td>0.0</td>
</tr>
<tr>
<td>RCO ‡</td>
<td>epitaxial region resistance</td>
<td>ohm</td>
<td>0.0</td>
</tr>
<tr>
<td>RE</td>
<td>emitter ohmic resistance</td>
<td>ohm</td>
<td>0.0</td>
</tr>
<tr>
<td>TF</td>
<td>ideal forward transit time</td>
<td>sec</td>
<td>0.0</td>
</tr>
<tr>
<td>TR</td>
<td>ideal reverse transit time</td>
<td>sec</td>
<td>0.0</td>
</tr>
<tr>
<td>TRB1</td>
<td>RB temperature coefficient (linear)</td>
<td>°C⁻¹</td>
<td>0.0</td>
</tr>
<tr>
<td>TRB2</td>
<td>RB temperature coefficient (quadratic)</td>
<td>°C⁻²</td>
<td>0.0</td>
</tr>
<tr>
<td>TRC1</td>
<td>RC temperature coefficient (linear)</td>
<td>°C⁻¹</td>
<td>0.0</td>
</tr>
<tr>
<td>TRC2</td>
<td>RC temperature coefficient (quadratic)</td>
<td>°C⁻²</td>
<td>0.0</td>
</tr>
<tr>
<td>TRE1</td>
<td>RE temperature coefficient (linear)</td>
<td>°C⁻¹</td>
<td>0.0</td>
</tr>
<tr>
<td>TRE2</td>
<td>RE temperature coefficient (quadratic)</td>
<td>°C⁻²</td>
<td>0.0</td>
</tr>
<tr>
<td>TRM1</td>
<td>RBM temperature coefficient (linear)</td>
<td>°C⁻¹</td>
<td>0.0</td>
</tr>
<tr>
<td>TRM2</td>
<td>RBM temperature coefficient (quadratic)</td>
<td>°C⁻²</td>
<td>0.0</td>
</tr>
<tr>
<td>T_ABS</td>
<td>absolute temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_MEASURED</td>
<td>measured temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_REL_GLOBAL</td>
<td>relative to current temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_REL_LOCAL</td>
<td>relative to AKO model temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>VAF (VA)</td>
<td>forward Early voltage</td>
<td>volt</td>
<td>infinite</td>
</tr>
<tr>
<td>VAR (VB)</td>
<td>reverse Early voltage</td>
<td>volt</td>
<td>infinite</td>
</tr>
<tr>
<td>VG</td>
<td>quasi-saturation extrapolated bandgap voltage at 0° K</td>
<td>V</td>
<td>1.206</td>
</tr>
<tr>
<td>VJC (PC)</td>
<td>base-collector built-in potential</td>
<td>volt</td>
<td>0.75</td>
</tr>
<tr>
<td>VJE (PE)</td>
<td>base-emitter built-in potential</td>
<td>volt</td>
<td>0.75</td>
</tr>
</tbody>
</table>
The parameters $I_{SE}(C_2)$ and $I_{SC}(C_4)$ can be set to be greater than one. In this case, they are interpreted as multipliers of $I_S$ instead of absolute currents: that is, if $I_{SE}$ is greater than one, then it is replaced by $I_{SE} \cdot I_S$. Likewise for $I_{SC}$.

If the model parameter $R_{CO}$ is specified, then quasi-saturation effects are included.

The distribution of the CJC capacitance is specified by $X_{CJC}$ and $X_{CJC2}$. The model parameter $X_{CJC2}$ is used like $X_{CJC}$. The differences between the two parameters are as follows.

When $X_{CJC2}$ is specified in the range $0 < X_{CJC2} < 1.0$, $X_{CJC}$ is ignored. Also, the extrinsic base to extrinsic collector capacitance ($C_{bx2}$) and the gain-bandwidth product ($F_t2$) are included in the operating point information (in the output listing generated during a Bias Point Detail analysis, OP (bias point)). For backward compatibility, the parameter $X_{CJC}$ and the associated calculation of $C_{bx}$ and $F_t$ remain unchanged. $C_{bx}$ and $F_t$ appears in the output listing only when $X_{CJC}$ is specified.

The use of $X_{CJC2}$ produces more accurate results because $C_{bx2}$ (the fraction of $C_{JC}$ associated with the intrinsic collector node) now equals the ratio of the device’s emitter area-to-base area. This results in a better correlation between the measured data and the gain bandwidth product ($F_t2$) calculated by PSpice.

$X_{CJS}$, which is valid in the range $0 \leq X_{CJS} \leq 1.0$, specifies a portion of the $C_{JS}$ capacitance to be between the external substrate and external collector nodes instead of between the external substrate and internal collector nodes. When $X_{CJS}$ is 1, $C_{JS}$ is applied totally between the external substrate and internal collector nodes. When $X_{CJS}$ is 0, $C_{JS}$ is applied totally between the external substrate and external collector codes.
Bipolar transistor equations

The equations in this section describe an NPN transistor. For the PNP and LPNP devices, reverse the signs of all voltages and currents.

The following variables are used:

- \( V_{be} \) = intrinsic base-intrinsic emitter voltage
- \( V_{bc} \) = intrinsic base-intrinsic collector voltage
- \( V_{bs} \) = intrinsic base-substrate voltage
- \( V_{bw} \) = intrinsic base-extrinsic collector voltage (quasi-saturation only)
- \( V_{bx} \) = extrinsic base-intrinsic collector voltage
- \( V_{ce} \) = intrinsic collector-intrinsic emitter voltage
- \( V_{js} \) = (NPN) intrinsic collector-substrate voltage
  \( = (\text{PNP}) \) intrinsic substrate-collector voltage
  \( = (\text{LPNP}) \) intrinsic base-substrate voltage
- \( V_{t} = k \cdot T/q \) (thermal voltage)
- \( k \) = Boltzmann’s constant
- \( q \) = electron charge
- \( T \) = analysis temperature (°K)
- \( T_{nom} \) = nominal temperature (set using the TNOM option)

Other variables are listed in **Bipolar transistor model parameters**.

Positive current is current flowing into a terminal.
Bipolar transistor equations for DC current

\( I_b = \) base current = \( \text{area} \cdot (I_{be1}/K_{q1} + I_{be2} + I_{bc1}/K_{q2} + I_{bc2}) \)

\( I_c = \) collector current = \( \text{area} \cdot (I_{be1}/K_{q1} - I_{bc1}/K_{q2} - I_{bc2}) \)

- \( I_{be1} = \) forward diffusion current = \( \text{IS} \cdot (e^{V_{be}/(N\cdot V_t)} - 1) \)
- \( I_{be2} = \) non-ideal base-emitter current = \( \text{ISE} \cdot (e^{V_{be}/(N\cdot V_t)} - 1) \)
- \( I_{bc1} = \) reverse diffusion current = \( \text{IS} \cdot (e^{V_{bc}/(N\cdot V_t)} - 1) \)
- \( I_{bc2} = \) non-ideal base-collector current = \( \text{ISC} \cdot (e^{V_{bc}/(N\cdot V_t)} - 1) \)
- \( K_{qb} = \) base charge factor = \( K_{q1} \cdot (1 + (1 + 4 \cdot K_{q2})^{-N})/2 \)

\( K_{q1} = 1/(1 - V_{bc}/V_{AF} - V_{be}/V_{AR}) \)

\( K_{q2} = I_{be1}/I_{KF} + I_{bc1}/I_{KR} \)

\( I_s = \) substrate current = \( \text{area} \cdot \text{ISS} \cdot (e^{V_{js}/(N\cdot V_t)} - 1) \)

\( R_b = \) actual base parasitic resistance

**Case 1**

for: \( IRB = \) infinite (default value)

then: \( R_b = (R_{BM} + (R_{B} - R_{BM})/K_{qb})/\text{area} \)

**Case 2**

For: \( IRB > 0 \)

then:
\[
R_b = (R_{BM} + 3 \cdot (R_{B} - R_{BM}) \cdot \frac{\tan(x) - x}{x \cdot (\tan(x))^2})/\text{area}
\]

where:
\[
x = \frac{(1 + (144/\pi^2) \cdot I_b/(\text{area} \cdot IRB))^{1/2} - 1}{(24/\pi^2) \cdot (I_b/(\text{area} \cdot IRB))^{1/2}}
\]
## Bipolar transistor equations for capacitance

All capacitances, except $C_{bx}$, are between terminals of the intrinsic transistor which is inside of the collector, base, and emitter parasitic resistances. $C_{bx}$ is between the intrinsic collector and the extrinsic base.

### base-emitter capacitance

- $C_{be} =$ base-emitter capacitance = $C_{tbe} + area \cdot C_{jbe}$
- $C_{tbe} =$ transit time capacitance = $t_f \cdot G_{be}$
  - $t_f =$ effective $T_f = T_f \cdot (1 + X_{TF} \cdot I_{be1}/(I_{be1} + area \cdot I_{TF}))$\(^2 \cdot e^{V_{bc}/(1.44 \cdot V_{TF})}\)
  - $G_{be} =$ DC base-emitter conductance = $(dI_{be})/(dV_b)$
  - $I_{be} =$ $I_{be1} + I_{be2}$

  $C_{jbe} = C_{JE} \cdot (1 - V_{be}/V_{JE})^{M_{JE}} \quad I F \quad V_{be} \leq F_C \cdot V_{JE}$

  $C_{jbe} = C_{JE} \cdot (1 - FC \cdot (1 + M_{JE}) + M_{JE} \cdot V_{be}/V_{JE}) \quad I F \quad V_{be} > F_C \cdot V_{JE}$

### base-collector capacitance

- $C_{bc} =$ base-collector capacitance = $C_{tbc} + area \cdot X_{JC} \cdot C_{jbc}$
- $C_{tbc} =$ transit time capacitance = $T_R \cdot G_{bc}$
  - $G_{bc} =$ DC base-collector conductance = $(dI_{bc})/(dV_{bc})$

  $C_{jbc} = C_{JC} \cdot (1 - V_{bc}/V_{JC})^{M_{JC}} \quad I F \quad V_{bc} \leq F_C \cdot V_{JC}$

  $C_{jbc} = C_{JC} \cdot (1 - FC \cdot (1 + M_{JC}) + M_{JC} \cdot V_{bc}/V_{JC}) \quad I F \quad V_{bc} > F_C \cdot V_{JC}$

### extrinsic-base to intrinsic-collector capacitance

- $C_{bx} =$ extrinsic-base to intrinsic-collector capacitance = $area \cdot (1 - X_{JC}) \cdot C_{jbx}$

  $C_{jbx} = C_{JC} \cdot (1 - V_{bx}/V_{JC})^{M_{JC}} \quad I F \quad V_{bx} \leq F_C \cdot V_{JC}$

  $C_{jbx} = C_{JC} \cdot (1 - FC (1 + M_{JC}) + M_{JC} \cdot V_{bx}/V_{JC}) \quad I F \quad V_{bx} > F_C \cdot V_{JC}$

### substrate junction capacitance

- $C_{js} =$ substrate junction capacitance = $area \cdot C_{jjs}$

  $C_{jjs} = C_{JS} \cdot (1 - V_{js}/V_{JS})^{M_{JS}} (\text{assumes } F_C = 0) \quad I F \quad V_{js} \leq 0$

  $C_{jjs} = C_{JS} \cdot (1 + M_{JS} \cdot V_{js}/V_{JS}) \quad I F \quad V_{js} > 0$
Bipolar transistor equations for quasi-saturation effect

Quasi-saturation is an operating region where the internal base-collector metallurgical junction is forward biased, while the external base-collector terminal remains reverse biased. This effect is modeled by extending the intrinsic Gummel-Poon model, adding a new internal node, a controlled current source, Iepi, and two controlled capacitances, represented by the charges Qo and Qw. These additions are only included if the model parameter $RCO$ is specified. See reference [3] of References for the derivation of this extension.

\[
I_{epi} = area \cdot \left( V_O \cdot (V_t \cdot (K(V_{bc}) - K(V_{bn}) - \ln((1+K(V_{bc}))/((1+K(V_{bn})))) + V_{bc} - V_{bn})) / R_{CO} \cdot (|V_{bc} - V_{bn}| + V_O) \right)
\]

\[
Q_o = area \cdot Q_{CO} \cdot (K(V_{bc}) - 1 - \text{GAMMA}/2)
\]

\[
Q_w = area \cdot Q_{CO} \cdot (K(V_{bn}) - 1 - \text{GAMMA}/2)
\]

where

\[
K(v) = (1 + \text{GAMMA} \cdot e^{v/V_t})^{1/2}
\]
The development of the temperature dependencies for the quasi-saturation model parameters \( \Gamma \), \( R_C \), and \( V_O \) are described in reference [3] on page 214. These temperature dependencies are only used when the model parameter \( QUASIMOD = 1.0 \).

\[
\Gamma(T) = \Gamma(T_{nom}) \cdot (T/T_{nom})^3 \cdot \exp(-q V_G/k \cdot (1/T - 1/T_{nom}))
\]
\[
R_C(T) = R_C(T_{nom}) \cdot (T/T_{nom})^{CN}
\]
\[
V_O(T) = V_O(T_{nom}) \cdot (T/T_{nom})^{CN \cdot D}
\]
Bipolar transistor equations for noise

Noise is calculated assuming a 1.0-hertz bandwidth, using the following spectral power densities (per unit bandwidth):

<table>
<thead>
<tr>
<th>parasitic resistances thermal noise</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RC</td>
<td>$I_c^2 = 4 \cdot k \cdot T / (R_C/\text{area})$</td>
</tr>
<tr>
<td>RB</td>
<td>$I_b^2 = 4 \cdot k \cdot T / R_B$</td>
</tr>
<tr>
<td>RE</td>
<td>$I_e^2 = 4 \cdot k \cdot T / (R_E/\text{area})$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>base and collector currents shot and flicker noise</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IB</td>
<td>$I_b^2 = 2 \cdot q \cdot I_b + K F \cdot I_b \cdot \text{AF} / \text{FREQUENCY}$</td>
</tr>
<tr>
<td>IC</td>
<td>$I_c^2 = 2 \cdot q \cdot I_c$</td>
</tr>
</tbody>
</table>

References

For a more information on bipolar transistor models, refer to:

[1] Ian Getreu, Modeling the Bipolar Transistor, Tektronix, Inc. part# 062-2841-00.

For a generally detailed discussion of the U.C. Berkeley SPICE models, including the bipolar transistor, refer to:


For a description of the extension for the quasi-saturation effect, refer to:

Resistor

General form

R<name> <+node> <-node> [model name] <value>
+ [TC = <TC1> [,<TC2>]]

Examples

RLOAD 15 0 2K
R2 1 2 2.4E4 TC=.015,-.003
RFDBCK 3 33 RMOD 10K

Model form

.MODEL <model name> RES [model parameters]

Arguments and options

(+ and (-) nodes

Define the polarity when the resistor has a positive voltage across it.

[model name]

Affects the resistance value; see Resistor value formulas.

Comments

The first node listed (or pin 1 in Capture) is defined as positive. The voltage across the component is therefore defined as the first node voltage minus the second node voltage.

Positive current flows from the (+) node through the resistor to the (-) node. Current flow from the first node through the component to the second node is considered positive.

Temperature coefficients for the resistor can be specified in-line, as in the second example. If the resistor has a model specified, then the coefficients from the model are used for the temperature updates; otherwise, the in-line values are used. In both cases the temperature coefficients have default values of zero. Expressions cannot be used for the in-line coefficients.

Capture parts

For standard R parts, the effective value of the part is set directly by the VALUE property. For the variable resistor, R_VAR, the effective value is the product of the base value (VALUE) and multiplier (SET).

In general, resistors should have positive component values (VALUE property). In all cases, components must not be given a value of zero.

However, there are cases when negative component values are desired. This occurs most often in filter designs that analyze an RLC circuit equivalent to a real circuit. When transforming from the real to the RLC equivalent, it is possible to end up with negative component values.
PSpice A/D allows negative component values for bias point, DC sweep, AC, and noise analyses. In the case of resistors, the noise contribution from negative component values come from the absolute value of the component (components are not allowed to generate negative noise). A transient analysis may fail for a circuit with negative components. Negative components may create instabilities in time that the analysis cannot handle.

<table>
<thead>
<tr>
<th>Part name</th>
<th>Model type</th>
<th>Property</th>
<th>Property description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>resistor</td>
<td>VALUE</td>
<td>resistance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TC</td>
<td>linear and quadratic temperature coefficients</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TOLERANCE</td>
<td>device tolerance (see page 52)</td>
</tr>
<tr>
<td>R_VAR</td>
<td>variable resistor</td>
<td>VALUE</td>
<td>base resistance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET</td>
<td>multiplier</td>
</tr>
</tbody>
</table>

The RBREAK part must be used if you want a LOT tolerance. In that case, use the Model Editor to edit the RBREAK instance.

**Breakout parts**

For non-stock passive and semiconductor devices, Capture has a set of breakout parts designed for customizing model parameters for simulation. These are useful for setting up Monte Carlo and worst-case analyses with device and/or lot tolerances specified for individual model parameters.

Basic breakout part names consist of the intrinsic PSpice A/D device letter plus the suffix BREAK. By default, the model name is the same as the part name and references the appropriate device model with all parameters set at their default. For instance, the DBREAK part references the DBREAK model, which is derived from the intrinsic PSpice A/D D model (.MODEL DBREAK D). Another approach is to use the model editor to derive an instance model and customize this. For example, you could add device and/or lot tolerances to model parameters.

For breakout part RBREAK, the effective value is computed from a formula that is a function of the specified VALUE property.

<table>
<thead>
<tr>
<th>Device type</th>
<th>Part name</th>
<th>Part library file</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resistor</td>
<td>RBREAK</td>
<td>BREAKOUT.OLB</td>
<td>VALUE</td>
<td>resistance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MODEL</td>
<td>RES model name</td>
</tr>
</tbody>
</table>
# Resistor model parameters

<table>
<thead>
<tr>
<th>Model parameters *</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>resistance multiplier</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>TC1</td>
<td>linear temperature coefficient</td>
<td>°C⁻¹</td>
<td>0.0</td>
</tr>
<tr>
<td>TC2</td>
<td>quadratic temperature coefficient</td>
<td>°C⁻²</td>
<td>0.0</td>
</tr>
<tr>
<td>TCE</td>
<td>exponential temperature coefficient</td>
<td>%/°C</td>
<td>0.0</td>
</tr>
<tr>
<td>T_ABS</td>
<td>absolute temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_MEASURED</td>
<td>measured temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_REL_GLOBAL</td>
<td>relative to current temperature</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>T_REL_LOCAL</td>
<td>relative to AKO model temperature</td>
<td>°C</td>
<td></td>
</tr>
</tbody>
</table>

* For information on T_MEASURED, T_ABS, T_REL_GLOBAL, and T_REL_LOCAL, see [MODEL (model definition)](model-definition).
Resistor equations

Resistor value formulas

One  If [model name] is included and TCE is specified, then the resistance is given by:

<value>·R·1.01 TCE·(T-Tnom)

where <value> is normally positive (though it can be negative, but not zero). Tnom is the nominal temperature (set using TNOM option).

Two  If [model name] is included and TCE is not specified, then the resistance is given by:

<value>·R·(1+ TC1·(T-Tnom)+ TC2·(T-Tnom)^2)

where <value> is usually positive (though it can be negative, but not zero).

Resistor equation for noise

Noise is calculated assuming a 1.0-hertz bandwidth. The resistor generates thermal noise using the following spectral power density (per unit bandwidth):

i^2 = 4·k·T/resistance
Voltage-controlled switch

General form
S<name> <(+ switch node) <(- switch node)
+ <(+ controlling node) <(- controlling node)
+ <model name>

Examples
S12   13 17  2 0 SMOD
SESET 5 0 15 3 RELAY

Model form
.CENTER <model name> VSWITCH [model parameters]

Description
The voltage-controlled switch is a special kind of voltage-controlled resistor. This switch model was designed to minimize numerical problems. However, there are a few things to consider; see Special considerations.

Comments
The resistance between the <(+ switch node) and <(- switch node) depends on the voltage between the <(+ controlling node) and <(- controlling node). The resistance varies continuously between the RON and ROFF model parameters.

A resistance of 1/GMIN is connected between the controlling nodes to keep them from floating. See the .OPTIONS (analysis options) statement for setting GMIN.

Although very little computer time is required to evaluate switches, during transient analysis the simulator must step through the transition region using a fine enough step size to get an accurate waveform. Applying many transitions can produce long run times when evaluating the other devices in the circuit at each time step.
Capture parts

Ideal switches

Summarized below is the available voltage-controlled switch part type in the `breakout.slb` part library. To create a time-controlled switch, connect the switch control pins to a voltage source with the appropriate voltage vs. time values (transient specification).

<table>
<thead>
<tr>
<th>Part type</th>
<th>Part Name</th>
<th>Model type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage-Controlled Switch</td>
<td>SBREAK</td>
<td>VSWITCH</td>
</tr>
</tbody>
</table>

The VSWITCH model defines the on/off resistance and the on/off control voltage or current thresholds. This switch has a finite on resistance and off resistance, and it changes smoothly between the two as its control voltage (or current) changes. This behavior is important because it allows PSpice A/D to find a continuous set of solutions for the simulation. You can make the on resistance very small in relation to the other circuit impedances, and you can make the off resistance very large in relation to the other circuit impedances.

Voltage-controlled switch model parameters

<table>
<thead>
<tr>
<th>Model Parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROFF **</td>
<td>off resistance</td>
<td>ohm</td>
<td>1E+6</td>
</tr>
<tr>
<td>RON</td>
<td>on resistance</td>
<td>ohm</td>
<td>1.0</td>
</tr>
<tr>
<td>VOFF</td>
<td>control voltage for off state</td>
<td>volt</td>
<td>0.0</td>
</tr>
<tr>
<td>VON</td>
<td>control voltage for on state</td>
<td>volt</td>
<td>1.0</td>
</tr>
</tbody>
</table>


** RON and ROFF must be greater than zero and less than 1/GMIN.

Special considerations

- Using double precision numbers, the simulator can only handle a dynamic range of about 12 decades. Making the ratio of ROFF to RON greater than 1E+12 is not recommended.

- Also, it not recommend to make the transition region too narrow. Remember that in the transition region the switch has gain. The narrower the region, the higher the gain and the greater the potential for numerical problems. The smallest allowed value for $|VON - VOFF|$ is $\text{RELTOL} \cdot (\max(|VON|, |VOFF|) + VNTOL)$. 
Voltage-controlled switch equations

In the following equations:

- $V_c$ = voltage across control nodes
- $L_m$ = log-mean of resistor values = $\ln((R_{ON} \cdot R_{OFF})^{1/2})$
- $L_r$ = log-ratio of resistor values = $\ln(R_{ON}/R_{OFF})$
- $V_m$ = mean of control voltages = $(V_{ON} + V_{OFF})/2$
- $V_d$ = difference of control voltages = $V_{ON} - V_{OFF}$
- $k$ = Boltzmann’s constant
- $T$ = analysis temperature (°K)
Voltage-controlled switch equations for switch resistance

Rs = switch resistance

For: VON > VOFF

if: $V_c \geq V_{ON}$
then: $Rs = R_{ON}$

if: $V_c \leq V_{OFF}$
then: $Rs = R_{OFF}$

if: $V_{OFF} < V_c < V_{ON}$
then: 
$$Rs = \exp(L_m + 3 \cdot L_r \cdot (V_c - V_m)/(2 \cdot V_d) - 2 \cdot L_r \cdot (V_c - V_m)^3 /V_d^3)$$

For: VON < VOFF

if: $V_c < V_{ON}$
then: $Rs = R_{ON}$

if: $V_c > V_{OFF}$
then: $Rs = R_{OFF}$

if: $V_{OFF} > V_c > V_{ON}$
then: 
$$Rs = \exp(L_m - 3 \cdot L_r \cdot (V_c - V_m)/(2 \cdot V_d) + 2 \cdot L_r \cdot (V_c - V_m)^3 /V_d^3)$$

Voltage-controlled switch equation for noise

Noise is calculated assuming a 1.0-hertz bandwidth. The voltage-controlled switch generates thermal noise as if it were a resistor having the same resistance that the switch has at the bias point, using the following spectral power density (per unit bandwidth):

$$i^2 = 4 \cdot k \cdot T / Rs$$
Transmission line

Description
The transmission line device is a bidirectional delay line with two ports, A and B. The (+) and (-) nodes define the polarity of a positive voltage at a port.

Comments
During transient (.TRAN (transient analysis)) analysis, the internal time step is limited to be no more than one-half the smallest transmission delay, so short transmission lines cause long run times.

The simulation status window displays the properties of the three shortest transmission lines in a circuit if a transient run’s time step ceiling is set more frequently by one of the transmission lines. This is helpful when you have a large number of transmission lines. The properties displayed are:

• % loss: percent attenuation at the characteristic delay (i.e., the degree to which the line is lossy)
• time step ceiling: induced by the line
• % of line delay: time step size at percentage of characteristic delay

These transmission line properties are displayed only if they are slowing down the simulation.

For a line that uses a model, the electrical length is given after the model name. Example T5 of Examples uses TMOD to specify the line parameters and has an electrical length of one unit.

All of the transmission line parameters from either the ideal or lossy parameter set can be expressions. In addition, R and G can be general Laplace expressions. This allows the user to model frequency dependent effects, such as skin effect and dielectric loss. However, this adds to the computation time for transient analysis, since the impulse responses must be obtained by an inverse FFT instead of analytically.
**Ideal line**

**General form**

\[ T\text{name} \text{<A port (+) node>} \text{<A port (-) node>} \\
+ \text{<B port (+) node>} \text{<B port (-) node>} \\
+ \text{[model name]} \\
+ \text{Z0=<value>} \text{[TD=<value>] [F=<value> [NL=<value>]]} \\
+ \text{IC= <near voltage> <near current> <far voltage> <far current>}

**Description**

As shown below, port A’s (+) and (-) nodes are 1 and 2, and port B’s (+) and (-) nodes are 3 and 4, respectively.

**Comments**

For the ideal line, IC sets the initial guess for the voltage or current across the ports. The <near voltage> value is the voltage across A(+) and A(-) and the <far voltage> is the voltage across B(+) and B(-). The <near current> is the current through A(+) and A(-) and the <far current> is the current through B(+) and B(-).

For the ideal case, Z0 is the characteristic impedance. The transmission line’s length can be specified either by TD, a delay in seconds, or by F and NL, a frequency and a relative wavelength at F. NL has a default value of 0.25 (F is the quarter-wave frequency). Although TD and F are both shown as optional, one of the two must be specified.

Both Z0 (Z-zero) and ZO (Z-O) are accepted by the simulator.
Lossy line

General form

\[ T\text{<name>} \ <A \text{ port } (+) \text{ node}> \ <A \text{ port } (-) \text{ node}> \]
+ \ <B \text{ port } (+) \text{ node}> \ <B \text{ port } (-) \text{ node}> 
+ [ \ <\text{model name}> \ [\text{electrical length value}] \ ]
+ LEN=<value> \ R=<value> \ L=<value>
+ G=<value> \ C=<value>

Examples

T1 1 2 3 4 Z0=220 TD=115ns
T2 1 2 3 4 Z0=220 F=2.25MEG
T3 1 2 3 4 Z0=220 F=4.5MEG NL=0.5
T4 1 2 3 4 LEN=1 R=.311 L=.378u G=6.27u C=67.3p
T5 1 2 3 4 T=MOD 1

Model form

.Model <model name> TRN [model parameters]

Description

The simulator uses a distributed model to represent the properties of a lossy transmission line. That is, the line resistance, inductance, conductance, and capacitance are all continuously apportioned along the line’s length. A common approach to simulating lossy lines is to model these characteristics using discrete passive elements to represent small sections of the line. This is the lumped model approach, and it involves connecting a set of many small subcircuits in series as shown below:

This method requires that there is enough lumps to adequately represent the distributed character of the line, and this often results in the need for a large netlist and correspondingly long simulation times. The method also produces spurious oscillations near the natural frequencies of the lumped elements.

An additional extension allows systems of coupled transmission lines to be simulated. Transmission line coupling is specified using the K device. This is done in much the same way that coupling is specified for inductors. See the description of Transmission line coupling for further details.

The distributed model allows freedom from having to determine how many lumps are sufficient, and eliminates the spurious oscillations. It also allows lossy lines to be simulated in a fraction of the time necessary when using the lumped approach, for the same accuracy.

Comments

For a lossy line, LEN is the electrical length. R, L, G, and C are the per unit length values of resistance, inductance, conductance, and capacitance, respectively.

Example T4 specifies a lossy line one meter long. The lossy line model is similar to that of the ideal case, except that the delayed voltage and current values include terms which vary with frequency. These terms are computed in transient analysis using an impulse response convolution method, and the internal time step is limited by the time resolution required to accurately model the frequency characteristics of the line. As with ideal lines, short lossy lines cause long run times.
Capture parts

Ideal and lossy transmission lines

Listed below are the properties that you can set per instance of an ideal (T) or lossy (TLOSSY) transmission line. The parts contained in the TLINE.SLB part library contain a variety of transmission line types. Their part properties vary.

<table>
<thead>
<tr>
<th>Part name</th>
<th>Model type</th>
<th>Property</th>
<th>Property description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>transmission line</td>
<td>Z0</td>
<td>characteristic impedance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TD</td>
<td>transmission delay</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F</td>
<td>frequency for NL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NL</td>
<td>number of wavelengths or wave number</td>
</tr>
<tr>
<td>TLOSSY*</td>
<td>transmission line</td>
<td>LEN</td>
<td>electrical length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R</td>
<td>per unit length resistance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L</td>
<td>per unit length inductance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>G</td>
<td>per unit length conductance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C</td>
<td>per unit length capacitance</td>
</tr>
</tbody>
</table>

*Not available for Basics+ users.

PSpice A/D uses a distributed model to represent the properties of a lossy transmission line. That is, the line resistance, inductance, conductance, and capacitance are all continuously apportioned along the line’s length.

A common approach to simulating lossy lines is to model these characteristics using discreet passive elements to represent small sections of the line. This is the lumped model approach, and it involves connecting a set of many small subcircuits in series. This method requires that enough lumps exist to adequately represent the distributed characteristic of the line. This often results in the need for a large netlist and correspondingly long simulation time. The method also produces spurious oscillations near the natural frequencies of the lumped elements.

The distributed model used in PSpice A/D frees you from having to determine how many lumps are sufficient, and eliminates the spurious oscillations. It also allows lossy lines to be simulated with the same accuracy in a fraction of the time required by the lumped approach.

In addition, you can make R and G general Laplace expressions. This allows frequency dependent effects to be modeled, such as skin effect and dielectric loss.
**Coupled transmission lines**

Listed below are the properties that you can set per instance of a coupled transmission line part. The part library provides parts that can accommodate up to five coupled transmission lines. You can also create new parts that have up to ten coupled lines.

<table>
<thead>
<tr>
<th>Part name</th>
<th>Model type</th>
<th>Property</th>
<th>Property description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2COUPLED</td>
<td>coupled transmission line—symmetric</td>
<td>LEN</td>
<td>electrical length</td>
</tr>
<tr>
<td>T3COUPLED</td>
<td></td>
<td>R</td>
<td>per unit length resistance</td>
</tr>
<tr>
<td>T4COUPLED</td>
<td></td>
<td>L</td>
<td>per unit length inductance</td>
</tr>
<tr>
<td>T5COUPLED</td>
<td></td>
<td>G</td>
<td>per unit length conductance</td>
</tr>
</tbody>
</table>

| T2COUPLEDX*  | coupled transmission line—asymmetric | LEN      | electrical length                 |
| T3COUPLEDX   |                                          | R        | per unit length resistance        |
| T4COUPLEDX   |                                          | L        | per unit length inductance        |
| T5COUPLEDX   |                                          | G        | per unit length conductance       |
|              |                                          | C        | per unit length capacitance       |
|              |                                          | LM       | per unit length mutual inductance |
|              |                                          | CM       | per unit length mutual capacitance|

| KCOUPLE2      | transmission line coupling matrix       | T1       | name of first coupled line        |
|               |                                          | T2       | name of second coupled line       |
|               |                                          | LM       | per unit length mutual inductance |
|               |                                          | CM       | per unit length mutual capacitance|

| KCOUPLE3      | T1       | name of first coupled line        |
|               | T2       | name of second coupled line       |
|               | T3       | name of third coupled line        |
|               | LMij     | per unit length mutual inductance between line Ti and line Tj |
|               | CMij     | per unit length mutual capacitance between line Ti and line Tj |

*T2COUPLEDX is functionally identical to T2COUPLED. However, the T2COUPLEDX implementation uses the expansion of the subcircuit referenced by T2COUPLED.*
Simulating coupled lines

Use the K device to simulate coupling between transmission lines. Each of the coupled transmission line parts provided in the standard part library translate to K device and T device declarations in the netlist. PSpice A/D compiles a system of coupled lines by assembling capacitive and inductive coupling matrices from all of the K devices involving transmission lines. Though the maximum order for any one system is ten lines, there is no explicit limitation on the number of separate systems that may appear in one simulation.

The simulation model is accurate for:

- ideal lines
- low-loss lossy lines
- systems of homogeneous, equally spaced high-loss lines

For more information, see Transmission line coupling.

Simulation considerations

When simulating, transmission lines with short delays can create performance bottlenecks by setting the time step ceiling to a very small value.

If one transmission line sets the time step ceiling frequently, PSpice A/D reports the three lines with the shortest time step. The status window displays the percentage attenuation, step ceiling, and step ceiling as percentage of transmission line delay.

If your simulation is running reasonably fast, you can ignore this information and let the simulation proceed. If the simulation is slowed significantly, you may want to cancel the simulation and modify your design. If the line is lossy and shows negligible attenuation, model the line as ideal instead.
Transmission line model parameters

<table>
<thead>
<tr>
<th>Model parameters*</th>
<th>Description</th>
<th>Units **</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>for all transmission lines</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IC</td>
<td>Sets the initial condition and all four values must be entered. Four values are expected when IC is specified: the near-end voltage, the near-end current, the far-end voltage, and the far-end current, given in that order.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>for ideal transmission lines</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZO</td>
<td>characteristic impedance</td>
<td>ohms</td>
<td>none</td>
</tr>
<tr>
<td>TD</td>
<td>transmission delay</td>
<td>seconds</td>
<td>none</td>
</tr>
<tr>
<td>F</td>
<td>frequency for NL</td>
<td>Hz</td>
<td>none</td>
</tr>
<tr>
<td>NL</td>
<td>relative wavelength</td>
<td>none</td>
<td>0.25</td>
</tr>
<tr>
<td><strong>for lossy transmission lines</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>per unit length resistance</td>
<td>ohms/unit length</td>
<td>none</td>
</tr>
<tr>
<td>L</td>
<td>per unit length inductance</td>
<td>henries/unit length</td>
<td>none</td>
</tr>
<tr>
<td>G</td>
<td>per unit length conductance</td>
<td>mhos/unit length</td>
<td>none</td>
</tr>
<tr>
<td>C</td>
<td>per unit length capacitance</td>
<td>farads/unit length</td>
<td>none</td>
</tr>
<tr>
<td>LEN***</td>
<td>physical length</td>
<td>agrees with RLGCM</td>
<td>none</td>
</tr>
</tbody>
</table>

* See MODEL (model definition). The order is from the most commonly used to the least commonly used parameter.

** Any length units can be used, but they must be consistent. For instance, if LEN is in feet, then the units of R must be in ohms/foot.

*** A lossy line with R=G=0 and LEN=1 is equivalent to an ideal line with \( ZO = \sqrt{\frac{L}{C}} \) and \( TD = LEN \cdot \sqrt{L \cdot C} \).
References

For more information on how the lossy transmission line is implemented, refer to:

Independent voltage source & stimulus

The Independent Current Source & Stimulus (I) and the Independent Voltage Source & Stimulus (V) devices have the same syntax. See Independent current source & stimulus.
**Current-controlled switch**

**General form**
\[ W\text{name} \text{<(+ switch node)> <(- switch node)> + <controlling V device name> <model name>} \]

**Examples**
- \[ W12 \text{ 13 17 VC WMOD} \]
- \[ W\text{RESET 5 0 VRESET RELAY} \]

**Model form**
\[ .\text{MODEL <model name> ISWITCH [model parameters]} \]

**Description**
The current-controlled switch is a special kind of current-controlled resistor.

![Diagram of current-controlled switch](image)

This model was chosen for a switch to try to minimize numerical problems. However, there are a few things to consider; see Special considerations.

**Arguments and options**
- **<controlling V device name>**
  - The current that the resistance between the <(+ switch node) and <(- switch node)> depends on.

  **RON and ROFF**
  - Must be greater than zero and less than 1/GMIN. The resistance varies continuously between them.

**Comments**
A resistance of 1/GMIN is connected between the controlling nodes to keep them from floating. See .OPTIONS (analysis options) for information on setting GMIN.

Although very little computer time is required to evaluate switches, during transient analysis the simulator must step through the transition region using a fine enough step size to get an accurate waveform. Having many transitions can produce long run times when evaluating the other devices in the circuit for many times.
Capture parts

Ideal switches

Summarized below is the available current-controlled switch part type in the breakout.slb part library. To create a time-controlled switch, connect the switch control pins to a voltage source with the appropriate voltage vs. time values (transient specification).

<table>
<thead>
<tr>
<th>Device type</th>
<th>Part name</th>
<th>Model type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current-controlled switch</td>
<td>WBREAK</td>
<td>ISWITCH</td>
</tr>
</tbody>
</table>

The ISWITCH model defines the on/off resistance and the on/off control voltage or current thresholds. This switch has a finite on resistance and off resistance, and it changes smoothly between the two as its control voltage (or current) changes. This behavior is important because it allows PSpice A/D to find a continuous set of solutions for the simulation. You can make the on resistance very small in relation to the other circuit impedances, and you can make the off resistance very large in relation to the other circuit impedances.

As with current-controlled sources (F, FPOLY, H, and HPOLY), WBREAK contains a current-sensing voltage source. When netlisted, WBREAK generates two device declarations to the circuit file set:

- one for the controlled switch
- one for the independent current-sensing voltage source

If you want to create a new part for a current-controlled switch (with, for example, different on/off resistance and current threshold settings in the ISWITCH model), the TEMPLATE property must account for the additional current-sensing voltage source.
Current-controlled switch model parameters

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>I_OFF</td>
<td>control current for off state</td>
<td>amp</td>
<td>0.0</td>
</tr>
<tr>
<td>I_ON</td>
<td>control current for on state</td>
<td>amp</td>
<td>1E-3</td>
</tr>
<tr>
<td>R_OFF</td>
<td>off resistance</td>
<td>ohm</td>
<td>1E+6</td>
</tr>
<tr>
<td>R_ON</td>
<td>on resistance</td>
<td>ohm</td>
<td>1.0</td>
</tr>
</tbody>
</table>

* See .MODEL (model definition).

Special considerations

Using double precision numbers, the simulator can handle only a dynamic range of about 12 decades. Therefore, it is not recommended making the ratio of $R_{OFF}$ to $R_{ON}$ greater than 1.0E+12.

Similarly, it is also not recommended making the transition region too narrow. Remembering that in the transition region the switch has gain. The narrower the region, the higher the gain and the greater the potential for numerical problems. The smallest allowed value for $|I_{ON}-I_{OFF}|$ is $\text{RELTOL} \cdot (\text{MAX}(|I_{ON}|, |I_{OFF}|)) + \text{ABSTOL}$.

Current-controlled switch equations

In the following equations:

- $I_c$ = controlling current
- $L_m$ = log-mean of resistor values = $\ln((R_{ON} \cdot R_{OFF})^{1/2})$
- $L_r$ = log-ratio of resistor values = $\ln(R_{ON}/R_{OFF})$
- $I_m$ = mean of control currents = $I_{ON}+I_{OFF})/2$
- $I_d$ = difference of control currents = $I_{ON}-I_{OFF}$
- $k$ = Boltzmann’s constant
- $T$ = analysis temperature (°K)
Current-controlled switch equations for switch resistance

For: $I_{ON} > I_{OFF}$

if:
   $I_c > I_{ON}$
then:
   $R_s = R_{ON}$

if:
   $I_c < I_{OFF}$
then:
   $R_s = R_{OFF}$

if:
   $I_{OFF} < I_c < I_{ON}$
then:
   $R_s = \exp\left(L_m + 3\cdot L_r \cdot (I_c-I_m)/(2\cdot I_d) - 2\cdot L_r \cdot (I_c-I_m)^3/(I_d)^3\right)$

For: $I_{ON} < I_{OFF}$

if:
   $I_c < I_{ON}$
then:
   $R_s = R_{ON}$

if:
   $I_c > I_{OFF}$
then:
   $R_s = R_{OFF}$

if:
   $I_{OFF} < I_c < I_{ON}$
then:
   $R_s = \exp\left(L_m - 3\cdot L_r \cdot (I_c-I_m)/(2\cdot I_d) + 2\cdot L_r \cdot (I_c-I_m)^3/(I_d)^3\right)$

Current-controlled switch equation for noise

Noise is calculated assuming a 1.0-hertz bandwidth. The current-controlled switch generates thermal noise as if it were a resistor using the same resistance as the switch has at the bias point, using the following spectral power density (per unit bandwidth):

$i^2 = 4 \cdot k \cdot T / R_s$
Subcircuit instantiation

Purpose

This statement causes the referenced subcircuit to be inserted into the circuit using the given nodes to replace the argument nodes in the definition. It allows a block of circuitry to be defined once and then used in several places.

General form

X<name> [node]* <subcircuit name> [PARAMS: <<name> = <value>>]* + [TEXT: < <name> = <text value> >]*

Examples

X12 100 101 200 201 DIFFAMP
XBUFF 13 15 UNITAMP
XFOLLOW IN OUT VCC VEE OUT OPAMP
XFELT 1 2 FILTER PARAMS: CENTER=200kHz
X27 A1 A2 A3 Y PLD PARAMS: MNTYMXDLY=1
+ TEXT: JEDEC_FILE=MYJEDEC.JED
XNANDI 25 28 7 MYPWR MYGND PARAMS: IO_LEVEL=2

Arguments and options

<subcircuit name>

The name of the subcircuit’s definition. See .SUBCKT (subcircuit).

PARAMS:

Passes values into subcircuits as arguments and into expressions inside the subcircuit.

TEXT:

Passes text values into subcircuits and into text expressions inside the subcircuit.

Comments

There must be the same number of nodes in the call as in the subcircuit’s definition.

Subcircuit references can be nested; that is, a call can be given to subcircuit A, whose definition contains a call to subcircuit B. The nesting can be to any level, but must not be circular; for example, if subcircuit A’s definition contains a call to subcircuit B, then subcircuit B’s definition must not contain a call to subcircuit A.
IGBT

General form

Z\text{<name>}\text{<collector>}\text{<gate>}\text{<emitter>}\text{<model name>}
+ [\text{AREA}=<\text{value}>] [\text{WB}=<\text{value}>] [\text{AGD}=<\text{value}>]
+ [\text{KP}=<\text{value}>] [\text{TAU}=<\text{value}>]

Examples

ZDRIVE 1 4 2 IGBT A AREA=10.1u WB=91u AGD=5.1u KP=0.381
Z231 3 2 9 IGBT27

Model form

.MODEL <model name> NIGBT [model parameters]

Description

The equivalent circuit for the IGBT is shown below. It is modeled as an intrinsic device (not as a subcircuit) and contains five DC current components and six charge (capacitive) components. An overview of the model equations is included below. For a more detailed description of the defining equations see references [1] through [4] of References.
Capture parts

The following table lists the set of IGBT breakout parts designed for customizing model parameters for simulation. These are useful for setting up Monte Carlo and worst-case analyses with device and/or lot tolerances specified for individual model parameters.

<table>
<thead>
<tr>
<th>Part name</th>
<th>Model type</th>
<th>Property</th>
<th>Property description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZBREAKN</td>
<td>IGBT</td>
<td>AGD</td>
<td>gate-drain overlap area</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AREA</td>
<td>area of the device</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KP</td>
<td>MOS transconductance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TAU</td>
<td>ambipolar recombination lifetime</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WB</td>
<td>Metallurgical base width</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MODEL</td>
<td>NIGBT model name</td>
</tr>
</tbody>
</table>

Setting operating temperature

Operating temperature can be set to be different from the global circuit temperature by defining one of the model parameters: T_ABS, T_REL_GLOBAL, or T_REL_LOCAL. Additionally, model parameters can be assigned unique measurement temperatures using the T_MEASURED model parameter. For more information, see IGBT model parameters.
IGBT device parameters

The general form of the IGBT syntax allows for the specification of five device parameters. These device parameters and their associated default values are defined in previous table. The IGBT model parameters and their associated default values are defined in the table that follows. Model parameters can be extracted from data sheet information by using the OrCAD Model Editor. Also, a library of model parameters for commercially available IGBTs is supplied with the software.

The parameters AGD, AREA, KP, TAU, and WB are specified as both device and model parameters, and they cannot be used in a Monte Carlo analysis.

When specified as device parameters, the assigned values take precedence over those which are specified as model parameters. Also, as device parameters (but not as model parameters), they can be assigned a parameter value and used in conjunction with a .DC or .STEP analysis.

<table>
<thead>
<tr>
<th>Device parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGD</td>
<td>gate-drain overlap area</td>
<td>m²</td>
<td>5.0E-6</td>
</tr>
<tr>
<td>AREA</td>
<td>area of the device</td>
<td>m²</td>
<td>1.0E-5</td>
</tr>
<tr>
<td>KP</td>
<td>MOS transconductance</td>
<td>A/V²</td>
<td>0.38</td>
</tr>
<tr>
<td>TAU</td>
<td>ambipolar recombination lifetime</td>
<td>sec</td>
<td>7.1E-6</td>
</tr>
<tr>
<td>WB</td>
<td>metallurgical base width</td>
<td>m</td>
<td>9.0E-5</td>
</tr>
</tbody>
</table>
# IGBT model parameters

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGD</td>
<td>gate-drain overlap area</td>
<td>m²</td>
<td>5.0E-6</td>
</tr>
<tr>
<td>AREA</td>
<td>area of the device</td>
<td>m²</td>
<td>1.0E-5</td>
</tr>
<tr>
<td>BVF</td>
<td>avalanche uniformity factor</td>
<td>none</td>
<td>1.0</td>
</tr>
<tr>
<td>BVN</td>
<td>avalanche multiplication exponent</td>
<td>none</td>
<td>4.0</td>
</tr>
<tr>
<td>CGS</td>
<td>gate-source capacitance per unit area</td>
<td>F/cm²</td>
<td>1.24E-8</td>
</tr>
<tr>
<td>COXD</td>
<td>gate-drain oxide capacitance per unit area</td>
<td>F/cm²</td>
<td>3.5E-8</td>
</tr>
<tr>
<td>JSNE</td>
<td>emitter saturation current density</td>
<td>A/cm²</td>
<td>6.5E-13</td>
</tr>
<tr>
<td>KF</td>
<td>triode region factor</td>
<td>none</td>
<td>1.0</td>
</tr>
<tr>
<td>KP</td>
<td>MOS transconductance</td>
<td>A/V²</td>
<td>0.38</td>
</tr>
<tr>
<td>MUN</td>
<td>electron mobility</td>
<td>cm²/(V·s)</td>
<td>1.5E3</td>
</tr>
<tr>
<td>MUP</td>
<td>hole mobility</td>
<td>cm²/(V·s)</td>
<td>4.5E2</td>
</tr>
<tr>
<td>NB</td>
<td>base doping</td>
<td>1/cm³</td>
<td>2.1E14</td>
</tr>
<tr>
<td>TAU</td>
<td>ambipolar recombination lifetime</td>
<td>sec</td>
<td>7.1E-6</td>
</tr>
<tr>
<td>THETA</td>
<td>transverse field factor</td>
<td>1/V</td>
<td>0.02</td>
</tr>
<tr>
<td>VT</td>
<td>threshold voltage</td>
<td>V</td>
<td>4.7</td>
</tr>
<tr>
<td>VTD</td>
<td>gate-drain overlap depletion threshold</td>
<td>V</td>
<td>1.0E-3</td>
</tr>
<tr>
<td>WB</td>
<td>metallurgical base width</td>
<td>m</td>
<td>9.0E-5</td>
</tr>
</tbody>
</table>

* See `.MODEL (model definition)` statement.
IGBT equations

In the following equations:

- $I_{\text{mos}}$ = MOSFET channel current
- $I_T$ = anode current
- $I_{\text{css}}$ = steady-state (bipolar) collector current
- $I_{\text{bss}}$ = Steady-state base current
- $I_{\text{mult}}$ = avalanche multiplication current
- $R_b$ = conductivity modulated base resistance
- $b$ = ambipolar mobility ratio
- $D_p$ = diffusion coefficient for holes
- $W$ = quasi-neutral base width
- $Q_{\text{cb}}$ = instantaneous excess carrier base charge
- $Q_b$ = background mobile carrier charge
- $n_i$ = intrinsic carrier concentration
- $M$ = avalanche multiplication factor
- $I_{\text{gen}}$ = (bipolar) collector-base thermally generated current
- $\varepsilon_{\text{si}}$ = dielectric permittivity of silicon
- $q$ = electron charge
- $W_{\text{bcj}}$ = base (bipolar) to collector depletion width
IGBT equations for DC current

<table>
<thead>
<tr>
<th>MOSFET channel current</th>
</tr>
</thead>
</table>
| $I_{MOS} = \begin{cases} 
0 & \text{For } V_{gs} < V_T \\
K_F \cdot K_P \cdot \left( \frac{(V_{gs} - V_T) \cdot V_{ds} - \frac{K_F \cdot V_{ds}^2}{2}}{1 + \Theta \cdot (V_{gs} - V_T)} \right) & \text{For } V_{ds} \leq \frac{(V_{gs} - V_T) \cdot K_F}{K_F} \\
\frac{K_P \cdot (V_{gs} - V_T)^2}{2 \cdot (1 + \Theta \cdot (V_{gs} - V_T))} & \text{For } V_{ds} > \frac{(V_{gs} - V_T) \cdot K_F}{K_F}
\end{cases}$ |

<table>
<thead>
<tr>
<th>anode current: current through the resistor $R_b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_T = \frac{V_{Ce}}{R_b}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>steady-state collector current</th>
</tr>
</thead>
</table>
| $I_{css} = \begin{cases} 
0 & \text{For } V_{eb} \leq 0 \\
\left( \frac{1}{1 + b} \right) \cdot I_T + \left( \frac{b}{1 + b} \right) \cdot \left( \frac{4 \cdot D_p}{W^2} \right) \cdot Q_{eb} & \text{For } V_{eb} > 0
\end{cases}$ |

<table>
<thead>
<tr>
<th>steady-state base current</th>
</tr>
</thead>
</table>
| $I_{bss} = \begin{cases} 
0 & \text{For } V_{eb} \leq 0 \\
\frac{Q_{eb}}{\tau} + \left( \frac{Q_{eb}^2}{Q_B} \right) \cdot \left( \frac{4 \cdot N_B}{n_i^2} \right) \cdot (JSNE \cdot AREA) & \text{For } V_{eb} > 0
\end{cases}$ |

<table>
<thead>
<tr>
<th>avalanche multiplication current</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{mult} = (M - 1) \cdot (I_{mos} + I_{css}) + M \cdot I_{gen}$</td>
</tr>
</tbody>
</table>
IGBT equations for capacitance

**gate source**

\[ C_{gs} = CGS \]
\[ Q_{gs} = CGS \cdot V_{gs} \]

**drain source**

\[ C_{ds} = \frac{(\text{AREA} - \text{AGD}) \cdot \varepsilon_{si}}{W_{dsj}} \]
\[ Q_{ds} = q \cdot (\text{AREA} - \text{AGD}) \cdot NB \cdot W_{dsj} \]

where \( W_{dsj} = \left( \frac{2 \cdot \varepsilon_{si} \cdot (V_{ds} + 0.6)}{q \cdot NB} \right) \)

**gate drain**

For \( V_{ds} < V_{gs} - V_{TD} \)

\[ C_{dg} = \text{COXD} \]
\[ Q_{dg} = \text{COXD} \cdot V_{dg} \]

For \( V_{ds} \geq V_{gs} - V_{TD} \)

\[ C_{dgj} = \frac{\text{COXD} \cdot \varepsilon_{si} \cdot \text{AGD}}{W_{dgj}} \]
\[ Q_{dg} = \frac{q \cdot NB \cdot \varepsilon_{si} \cdot \text{AGD}}{\text{COXD}} \left( \frac{\text{COXD} \cdot W_{dgj}}{\varepsilon_{si} \cdot \text{AGD}} - \log \left( 1 + \frac{\text{COXD} \cdot W_{dgj}}{\varepsilon_{si} \cdot \text{AGD}} \right) \right) - \text{COXD} \cdot V_{TD} \]

where \( \frac{\text{COXD} \cdot \varepsilon_{si} \cdot \text{AGD}}{W_{dgj}} \)

**Ccer**

\[ C_{cer} = \frac{Q_{eb} \cdot C_{bcj}}{3 \cdot Q_{B}} \]
\[ C_{bcj} = \frac{\varepsilon_{si} \cdot \text{AREA}}{W_{bcj}} \]

**Cmult**

\[ C_{mult} = (M - 1) \cdot C_{cer} \]
\[ Q_{mult} = (M - 1) \cdot Q_{cer} \]

**emitter base**

\[ C_{eb} = \frac{dQ_{eb}}{dV_{eb}} \]
References

For more information on the IGBT model, refer to:


Digital devices

<table>
<thead>
<tr>
<th>Behavioral primitives</th>
<th>Multi-bit A/D and D/A converter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bidirectional transfer gates</td>
<td>Programmable logic array</td>
</tr>
<tr>
<td>Delay line</td>
<td>Pullup and pulldown</td>
</tr>
<tr>
<td>Digital input (N device)</td>
<td>Random access read-write memory</td>
</tr>
<tr>
<td>Digital output (O device)</td>
<td>Read only memory</td>
</tr>
<tr>
<td>File stimulus</td>
<td>Standard gates</td>
</tr>
<tr>
<td>Flip-flops and latches</td>
<td>Stimulus generator</td>
</tr>
<tr>
<td>Input/output model</td>
<td>Tristate gates</td>
</tr>
</tbody>
</table>
Digital device summary

<table>
<thead>
<tr>
<th>Device class</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>primitives</td>
<td>U</td>
<td>low-level digital devices (e.g., gates and flip-flops)</td>
</tr>
<tr>
<td>stimuli</td>
<td>U</td>
<td>digital stimulus generators</td>
</tr>
<tr>
<td></td>
<td></td>
<td>file-based stimulus</td>
</tr>
<tr>
<td>interface</td>
<td>N</td>
<td>digital input device</td>
</tr>
<tr>
<td></td>
<td>O</td>
<td>digital output device</td>
</tr>
</tbody>
</table>

Primitives are primarily used in subcircuits to model complete devices. Stimulus devices are used in the circuit to provide input for other digital devices during the simulation. Interface devices are mainly used inside subcircuits that model analog/digital and digital/analog interfaces.

The digital devices are part of the digital simulation feature of PSpice A/D. For more information on digital simulation and creating models, refer to your PSpice user’s guide.
Digital primitive summary

Digital primitives are low-level devices whose main use is modeling off-the-shelf parts, often in combination with each other.

Digital primitives should not be confused with the subcircuits in the libraries that use them. For instance, the 74LS00 subcircuit in `74ls.lib` uses a NAND digital primitive to model the 74LS00 part, but it also includes timing and interface information that makes the model adapted for use in a circuit simulation. For more information, refer to your PSpice user’s guide.

This section provides a reference for each of the digital primitives supported by the simulator, to help you create digital parts that are not in the model library.

<table>
<thead>
<tr>
<th>Primitive class</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standard gates</strong></td>
<td>BUF</td>
<td>buffer</td>
</tr>
<tr>
<td></td>
<td>INV</td>
<td>inverter</td>
</tr>
<tr>
<td></td>
<td>AND</td>
<td>AND gate</td>
</tr>
<tr>
<td></td>
<td>NAND</td>
<td>NAND gate</td>
</tr>
<tr>
<td></td>
<td>OR</td>
<td>OR gate</td>
</tr>
<tr>
<td></td>
<td>NOR</td>
<td>NOR gate</td>
</tr>
<tr>
<td></td>
<td>XOR</td>
<td>exclusive OR gate</td>
</tr>
<tr>
<td></td>
<td>NXOR</td>
<td>exclusive NOR gate</td>
</tr>
<tr>
<td></td>
<td>BUFA</td>
<td>buffer array</td>
</tr>
<tr>
<td></td>
<td>INVA</td>
<td>inverter array</td>
</tr>
<tr>
<td></td>
<td>ANDA</td>
<td>AND gate array</td>
</tr>
<tr>
<td></td>
<td>NANDA</td>
<td>NAND gate array</td>
</tr>
<tr>
<td></td>
<td>ORA</td>
<td>OR gate array</td>
</tr>
<tr>
<td></td>
<td>NORA</td>
<td>NOR gate array</td>
</tr>
<tr>
<td></td>
<td>XORA</td>
<td>exclusive OR gate array</td>
</tr>
<tr>
<td></td>
<td>NXORA</td>
<td>exclusive NOR gate array</td>
</tr>
<tr>
<td></td>
<td>AO</td>
<td>AND-OR compound gate</td>
</tr>
<tr>
<td></td>
<td>OA</td>
<td>OR-AND compound gate</td>
</tr>
<tr>
<td></td>
<td>AOI</td>
<td>AND-NOR compound gate</td>
</tr>
<tr>
<td></td>
<td>OAI</td>
<td>OR-NAND compound gate</td>
</tr>
<tr>
<td>Primitive class</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>--------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td><strong>Tristate gates</strong></td>
<td>BUF3</td>
<td>buffer</td>
</tr>
<tr>
<td></td>
<td>INV3</td>
<td>inverter</td>
</tr>
<tr>
<td></td>
<td>AND3</td>
<td>AND gate</td>
</tr>
<tr>
<td></td>
<td>NAND3</td>
<td>NAND gate</td>
</tr>
<tr>
<td></td>
<td>OR3</td>
<td>OR gate</td>
</tr>
<tr>
<td></td>
<td>NOR3</td>
<td>NOR gate</td>
</tr>
<tr>
<td></td>
<td>XOR3</td>
<td>exclusive OR gate</td>
</tr>
<tr>
<td></td>
<td>NXOR3</td>
<td>exclusive NOR gate</td>
</tr>
<tr>
<td></td>
<td>BUF3A</td>
<td>buffer array</td>
</tr>
<tr>
<td></td>
<td>INV3A</td>
<td>inverter array</td>
</tr>
<tr>
<td></td>
<td>AND3A</td>
<td>AND gate array</td>
</tr>
<tr>
<td></td>
<td>NAND3A</td>
<td>NAND gate array</td>
</tr>
<tr>
<td></td>
<td>OR3A</td>
<td>OR gate array</td>
</tr>
<tr>
<td></td>
<td>NOR3A</td>
<td>NOR gate array</td>
</tr>
<tr>
<td></td>
<td>XOR3A</td>
<td>exclusive OR gate array</td>
</tr>
<tr>
<td></td>
<td>NXOR3A</td>
<td>exclusive NOR gate array</td>
</tr>
<tr>
<td><strong>Bidirectional transfer gates</strong></td>
<td>NBTG</td>
<td>N-channel transfer gate</td>
</tr>
<tr>
<td></td>
<td>PBTG</td>
<td>P-channel transfer gate</td>
</tr>
<tr>
<td><strong>Flip-flops and latches</strong></td>
<td>JKFF</td>
<td>J-K, negative-edge triggered</td>
</tr>
<tr>
<td></td>
<td>DFF</td>
<td>D-type, positive-edge triggered</td>
</tr>
<tr>
<td></td>
<td>SRFF</td>
<td>S-R gated latch</td>
</tr>
<tr>
<td></td>
<td>DLTCH</td>
<td>D gated latch</td>
</tr>
<tr>
<td><strong>Pullup and pulldown</strong></td>
<td>PULLUP</td>
<td>pullup resistor array</td>
</tr>
<tr>
<td></td>
<td>PULLDN</td>
<td>pulldown resistor array</td>
</tr>
<tr>
<td><strong>Delay line</strong></td>
<td>DLYLINE</td>
<td>delay line</td>
</tr>
<tr>
<td><strong>Programmable logic array</strong></td>
<td>PLAND</td>
<td>AND array</td>
</tr>
<tr>
<td></td>
<td>PLOR</td>
<td>OR array</td>
</tr>
<tr>
<td></td>
<td>PLXOR</td>
<td>exclusive OR array</td>
</tr>
<tr>
<td></td>
<td>PLNAND</td>
<td>NAND array</td>
</tr>
<tr>
<td></td>
<td>PLNOR</td>
<td>NOR array</td>
</tr>
<tr>
<td></td>
<td>PLNXOR</td>
<td>exclusive NOR array</td>
</tr>
<tr>
<td></td>
<td>PLANDC</td>
<td>AND array, true and complement</td>
</tr>
<tr>
<td></td>
<td>PLORC</td>
<td>OR array, true and complement</td>
</tr>
<tr>
<td></td>
<td>PLXORC</td>
<td>exclusive OR array, true and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>complement</td>
</tr>
<tr>
<td></td>
<td>PLNANDC</td>
<td>NAND array, true and complement</td>
</tr>
<tr>
<td></td>
<td>PLNORC</td>
<td>NOR array, true and complement</td>
</tr>
<tr>
<td></td>
<td>PLNXORC</td>
<td>exclusive NOR array, true and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>complement</td>
</tr>
</tbody>
</table>
Digital devices

**Digital primitive summary**

<table>
<thead>
<tr>
<th>Primitive class</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read only memory</td>
<td>ROM</td>
<td>read-only memory</td>
</tr>
<tr>
<td>Random access read-write memory</td>
<td>RAM</td>
<td>random access read-write memory</td>
</tr>
<tr>
<td>Multi-bit A/D and D/A converter</td>
<td>ADC</td>
<td>multi-bit A/D converter</td>
</tr>
<tr>
<td></td>
<td>DAC</td>
<td>multi-bit D/A converter</td>
</tr>
<tr>
<td>Behavioral primitives</td>
<td>LOGICEXP</td>
<td>logic expression</td>
</tr>
<tr>
<td></td>
<td>PINDLY</td>
<td>pin-to-pin delay</td>
</tr>
<tr>
<td></td>
<td>CONSTRAN</td>
<td>constraint checking</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>

The format for specifying a digital primitive follows the general format described in the next section. Primitive-specific formats are also described which includes parameters and nodes that are specific to the primitive type.

Also listed is the specific timing model format for each primitive, along with the appropriate timing model parameters.

For example, the 74393 part provided in the model library is defined as a subcircuit composed of U devices as shown below.

```
subckt 74393 A CLR QA QB QC QD
+ optional: DPWR=$G_DPWR DGND=$G_DGND
+ params: MNTYMXDLY=0 IO_LEVEL=0
UINV inv DPWR DGND
+ CLR CLRBAR
+ D0_GATE IO_STD IO_LEVEL={IO_LEVEL}
U1 jkff(1) DPWR DGND
+ $D_HI CLRBAR A $D_HI $D_HI QA_BUF $D_NC
+ D_393_1 IO_STD MNTYMXDLY={MNTYMXDLY}=
+ IO_LEVEL={IO_LEVEL}
U2 jkff(1) DPWR DGND
+ $D_HI CLRBAR QA_BUF $D_HI $D_HI QB_BUF $D_NC
+ D_393_2 IO_STD MNTYMXDLY={MNTYMXDLY}=
U3 jkff(1) DPWR DGND
+ $D_HI CLRBAR QB_BUF $D_HI $D_HI QC_BUF $D_NC
+ D_393_3 IO_STD MNTYMXDLY={MNTYMXDLY}=
U4 jkff(1) DPWR DGND
+ $D_HI CLRBAR QC_BUF $D_HI $D_HI QD_BUF $D_NC
+ D_393_3 IO_STD MNTYMXDLY={MNTYMXDLY}=
UBUFF bufa(4) DPWR DGND
+ QA_BUF QB_BUF QC_BUF QD_BUF QA QB QC QD
+ D_393_4 IO_STD MNTYMXDLY={MNTYMXDLY}=
.ends
```

When adding digital parts to a part library, you can create corresponding digital device models by connecting U devices in a subcircuit definition similar to the one shown above. OrCAD recommends that these be saved in a custom model file. The model files can then be configured into the model library or specified for use in a given design.
## General digital primitive format

The format of digital primitives is similar to that of analog devices. One difference is that most digital primitives use two models instead of one. One of the models is the timing model, which specifies propagation delays and timing constraints, such as setup and hold times. The other model is the I/O model, which specifies information specific to the device’s input/output characteristics. The reason for having two models is that, while timing information is specific to a device, the input/output characteristics apply to a whole device family. Thus, many devices in the same family reference the same I/O model, but each device has its own timing model. If wanted, the timing models can be selected among primitives of the same class.

The general digital primitive format is shown below. Each statement can span one or more lines by using the `+ (line continuation)` character in the first column position. Comments can be added to each line by using the `; (in-line comment)`. For specific information on each primitive type, see the sections that follow.

### General form

```
U<name> <primitive type> [(<parameter value>*)]
+ <digital power node> <digital ground node>
+ <node>*
+ <timing model name> <I/O model name>
+ [MTYMXDLY=<delay select value>]
+ [IO_LEVEL=<interface subckt select value>]
```

### Model form

```
.MODEL <model name> UIO ( <model parameters>* )
```

See [Input/output model parameters](#) for a list of the UIO model parameters.

### Timing model format

```
.MODEL <model name> <model type> ( <model parameters>* )
```

### Examples

```
U1 NAND(2) $G_DPWR $G_DGND 1 2 10 D0_GATE IO_DFT
U2 JKFF(1) $G_DPWR $G_DGND 3 5 200 3 3 10 2 D_293ASTD IO_STD
U3 INV $G_DPWR $G_DGND IN OUT D_INV IO_INV MNTYMXDLY=3 IO_LEVEL=2
```

### Arguments and options

#### <primitive type> [(<parameter value>*)]

The type of digital device, such as NAND, JKFF, or INV. It is followed by zero or more parameters specific to the primitive type, such as number of inputs. The number and meaning of the parameters depends on the primitive type. See the sections that follow for a complete description of each primitive type and its parameters.

#### <digital power node> <digital ground node>

These nodes are used by the interface subcircuits which connect analog nodes to digital nodes or vice versa. Refer to your PSpice user’s guide for more information.

#### <node>*

One or more input and output nodes. The number of nodes depends on the primitive type and its parameters. Analog devices, digital devices, or both can be connected to a node. If a node has both analog and digital connections, then the simulator automatically inserts an interface subcircuit to translate between logic levels and voltages. Refer to your PSpice user’s guide for more information.
<timing model name>
The name of a timing model that describes the device’s timing characteristics, such as propagation delay and setup and hold times. Each timing parameter has a minimum, typical, or maximum value which can be selected using the optional `MNTYMXDLY` device parameter (described below) or the `DIGMNTYMX` option (see `.OPTIONS (analysis options)`). The type of the timing model and its parameters are specific to each primitive type and are discussed in the following sections. (Note that the PULLUP, PULLDN, and PINDLY primitives do not have timing models.)

<I/O model name>
The name of an I/O model, which describes the device’s loading and driving characteristics. I/O models also contain the names of up to four DtoA and AtoD interface subcircuits, which are automatically called by the simulator to handle interface nodes. Refer to your PSpice user’s guide for a more detailed description of I/O models.

<model type>
Is specific to the primitive type. See the specific primitive for the correct <model type> and associated <model parameters>. General timing model issues are discussed in the next section.

MNTYMXDLY
An optional device parameter that selects either the minimum, typical, or maximum delay values from the device’s timing model. A fourth option operates the primitive in Digital Worst-Case (min/max) mode. If not specified, MNTYMXDLY defaults to 0. Valid values are:

0 = Current value of `.OPTIONS DIGMNTYMX` (default=2)
1 = Minimum
2 = Typical
3 = Maximum
4 = Worst-case (min/max) timing

IO_LEVEL
An optional device parameter that selects one of the four AtoD or DtoA interface subcircuits from the device’s I/O model. The simulator calls the selected subcircuit automatically in the event a node connecting to the primitive also connects to an analog device. If not specified, IO_LEVEL defaults to 0. Valid values are:

0 = the current value of `.OPTIONS DIGIOLVL` (default=1)
1 = AtoD1/DtoA1
2 = AtoD2/DtoA2
3 = AtoD3/DtoA3
4 = AtoD4/DtoA4

Refer to your PSpice user’s guide for more information.
Timing models

With the exception of the PULLUP, PULLDN, and PINDLY devices, all digital primitives have a timing model that provides timing parameters to the simulator. Within a timing model, there can be one or more types of parameters

- propagation delays (TP)
- setup times (TSU)
- hold times (TH)
- pulse widths (TW)
- switching times (TSW)

Each parameter is further divided into three values: minimum (MN), typical (TY), and maximum (MX). For example, the typical low-to-high propagation delay on a gate is specified as $T_{PLH_{TY}}$. The minimum data-to-clock setup time on a flip-flop is specified as $T_{SU_{DCLK_{MN}}}$.

One or more parameters can be missing from the timing model definition. Data books do not always provide all three (minimum, typical, and maximum) timing specifications. The way the simulator handles missing parameters depends on the type of parameter.

Treatment of unspecified propagation delays

This discussion applies only to propagation delay parameters (TP). All other timing parameters, such as setup/hold times and pulse widths, are handled differently and are described in Treatment of unspecified timing constraints.

Often, only the typical and maximum delays are specified in data books. If, in this case, the simulator were to assume that the unspecified minimum delay just defaults to zero, the logic in certain circuits could break down.

For this reason, the simulator provides two configurable options, DIGMNTYSCALE and DIGTYMXSACLE (set using the .OPTIONS (analysis options) command), which are used to extrapolate unspecified propagation delays in the timing models.

**DIGMNTYSCALE**

This option computes the minimum delay when a typical delay is known, using the formula

$$TP_{xxMN} = DIGMNTYSCALE \cdot TP_{xxTY}$$

**DIGMNTYSCALE** has a default value of 0.4, or 40% of the typical delay. Its value must be between 0.0 and 1.0.

**DIGTYMXSACLE**

This option computes the maximum delay from a typical delay, using the formula

$$TP_{xxMX} = DIGTYMXSACLE \cdot TP_{xxTY}$$

**DIGTYMXSACLE** has a default value of 1.6. Its value must be greater than 1.0.
When a typical delay is unspecified, its value is derived from the minimum and/or maximum delays, in one of the following ways. If both the minimum and maximum delays are known, the typical delay is the average of these two values. If only the minimum delay is known, the typical delay is derived using the value of the DIGMNTYSCALE option. Likewise, if only the maximum delay is specified, the typical delay is derived using DIGTYMXSCALE. Obviously, if no values are specified, all three delays have a default value of zero.

Treatment of unspecified timing constraints

The remaining timing constraint parameters are handled differently from the propagation delays. Often, data books state pulse widths, setup times, and hold times as a minimum value. These parameters do not lend themselves to the extrapolation method used for propagation delays.

Instead, when one or more timing constraints are omitted, the simulator uses the following steps to fill in the missing values:

- If the minimum value is omitted, the default value is zero.
- If the maximum value is omitted, it takes on the typical value if one was specified, otherwise it takes on the minimum value.
- If the typical value is omitted, it is computed as the average of the minimum and maximum values.
Gates

Logic gates come in two types: standard and tristate. Standard gates always have their outputs enabled, whereas tristate gates have an enable control. When the enable control is 0, the output’s strength is Z and its level is X.

Logic gates also come in two forms: simple gates and gate arrays. Simple gates have one or more inputs and only one output. Gate arrays contain one or more simple gates in one component. Gate arrays allow one to work directly using parts that have several gates in one package.

The usual Boolean equations apply to these gates having the addition of the X level. The rule for X is: if an input is X, and if changing that input between one and zero would cause the output to change, then the output is also X. In other words, X is only propagated to the output when necessary. For example: 1 AND X = X; 0 AND X = 0; 0 OR X = X; 1 OR X = 1.
Standard gates

Device format

U<name> <gate type> (<parameter value>*)
+ <digital power node> <digital ground node>
+ <input node>* <output node>*
+ <timing model name> <I/O model name>
+ [MNTYMXDLY=<delay select value>]
+ [IO_LEVEL=<interface subckt select value>]

The standard gate types and their parameters are listed in Standard Gate Types.

Timing model format

<timing model name> UGATE [model parameters]

Examples

U5 AND(2) $G_DPWR $G_DGND IN0 IN1 OUT ; two-input AND gate
+ T_AND2 IO_STD

U2 INV $G_DPWR $G_DGND 3 5 ; simple INVerter
+ T_INV IO_STD

U13 NANDA(2,4) $G_DPWR $G_DGND ; four two-input NAND gates
+ INAO INA1 INB0 INB1 INC0 INC1
+ INDO INDI OUTA OUTB OUTC OUTD
+ T_NANDA IO_STD

U9 AO(3,3) $G_DPWR $G_DGND ;three-input AND-OR gate
+ INAO INA1 INA2 INB0 INB1 INB2 INC0 INC1 INC2
+ OUT T_AO IO_STD
+ MNTYMXDLY=1 IO_LEVEL=1

.Model T_AND2 UGATE ; AND2 Timing Model
+ TPLHMN=15ns TPLHTY=20ns TPLHMX=25ns
+ TPHLMN=10ns TPHLTY=15ns TPHLMX=20ns
+

Arguments and options

<no. of inputs><no. of gates>

The <no. of inputs> is the number of inputs per gate and <no. of gates> is the number of gates. in* and out* mean one or more nodes, whereas in and out refer to only one node.

In gate arrays the order of the nodes is: all inputs for the first gate, all inputs for the second gate, ..., output for the first gate, output for the second gate, ... In other words, all of the input nodes come first, then all of the output nodes. The total number of input nodes is <no. of inputs>-<no. of gates>; the number of output nodes is <no. of gates>.

A compound gate is a set of <no. of gates> first-level gates which each have <no. of inputs> inputs. Their outputs are connected to a single second-level gate. For example, the AO component has <no. of gates> AND gates whose outputs go into one OR gate. The OR gate’s output is the AO device’s output. The order of the nodes is: all inputs for the first, first-level gate; all inputs for the second, first-level gate; ...; the output of the second-level gate. In other words, all of the input nodes followed by the one output node.
### Standard gates

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameters</th>
<th>Nodes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>(&lt;no. of inputs&gt;)</td>
<td>in*, out</td>
<td>AND gate</td>
</tr>
<tr>
<td>ANDA</td>
<td>(&lt;no. of inputs&gt;,&lt;no. of gates&gt;)</td>
<td>in*, out*</td>
<td>AND gate array</td>
</tr>
<tr>
<td>AO</td>
<td>(&lt;no. of inputs&gt;,&lt;no. of gates&gt;)</td>
<td>in*, out</td>
<td>AND-OR compound gate</td>
</tr>
<tr>
<td>AOI</td>
<td>(&lt;no. of inputs&gt;,&lt;no. of gates&gt;)</td>
<td>in*, out*</td>
<td>AND-NOR compound gate</td>
</tr>
<tr>
<td>BUF</td>
<td>not applicable</td>
<td>in, out</td>
<td>buffer</td>
</tr>
<tr>
<td>BUFA</td>
<td>(&lt;no. of gates&gt;)</td>
<td>in*, out*</td>
<td>buffer array</td>
</tr>
<tr>
<td>INV</td>
<td>not applicable</td>
<td>in, out</td>
<td>inverter</td>
</tr>
<tr>
<td>INVA</td>
<td>(&lt;no. of gates&gt;)</td>
<td>in*, out*</td>
<td>inverter array</td>
</tr>
<tr>
<td>NAND</td>
<td>(&lt;no. of inputs&gt;)</td>
<td>in*, out</td>
<td>NAND gate</td>
</tr>
<tr>
<td>NANDA</td>
<td>(&lt;no. of inputs&gt;,&lt;no. of gates&gt;)</td>
<td>in*, out*</td>
<td>NAND gate array</td>
</tr>
<tr>
<td>NOR</td>
<td>(&lt;no. of inputs&gt;)</td>
<td>in*, out</td>
<td>NOR gate</td>
</tr>
<tr>
<td>NORA</td>
<td>(&lt;no. of inputs&gt;,&lt;no. of gates&gt;)</td>
<td>in*, out*</td>
<td>NOR gate array</td>
</tr>
<tr>
<td>NXOR</td>
<td>not applicable</td>
<td>in1, in2, out</td>
<td>exclusive NOR gate</td>
</tr>
<tr>
<td>NXORA</td>
<td>(&lt;no. of gates&gt;)</td>
<td>in*, out*</td>
<td>exclusive NOR gate array</td>
</tr>
<tr>
<td>OA</td>
<td>(&lt;no. of inputs&gt;,&lt;no. of gates&gt;)</td>
<td>in*, out</td>
<td>OR-AND compound gate</td>
</tr>
<tr>
<td>OAI</td>
<td>(&lt;no. of inputs&gt;,&lt;no. of gates&gt;)</td>
<td>in*, out*</td>
<td>OR-NAND compound gate</td>
</tr>
<tr>
<td>OR</td>
<td>(&lt;no. of inputs&gt;)</td>
<td>in*, out</td>
<td>OR gate</td>
</tr>
<tr>
<td>ORA</td>
<td>(&lt;no. of inputs&gt;,&lt;no. of gates&gt;)</td>
<td>in*, out*</td>
<td>OR gate array</td>
</tr>
<tr>
<td>XOR</td>
<td>not applicable</td>
<td>in1, in2, out</td>
<td>exclusive OR gate</td>
</tr>
<tr>
<td>XORA</td>
<td>(&lt;no. of gates&gt;)</td>
<td>in*, out*</td>
<td>exclusive OR gate array</td>
</tr>
</tbody>
</table>
## Standard gate timing model parameters

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPLHMN</td>
<td>delay: low to high, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPLHTY</td>
<td>delay: low to high, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPLHMX</td>
<td>delay: low to high, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPHLMN</td>
<td>delay: high to low, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPHLTY</td>
<td>delay: high to low, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPHLMX</td>
<td>delay: high to low, max</td>
<td>sec</td>
<td>0</td>
</tr>
</tbody>
</table>

* See [MODEL (model definition)](model-definition)
Tristate gates

Device format

U<name> <tristate gate type> [( <parameter value>* )]
+ <digital power node> <digital ground node>
+ <input node>* <enable node> <output node>*
+ <timing model name> <I/O model name>
+ [MNTYMXDLY=<delay select value>]
+ [IO_LEVEL=<interface subckt select value>]

Timing model format

.MODEL <timing model name> UTGATE [model parameters]

Examples

U5  AND3(2) $G_DPWR $G_DGND IN0 IN1 ENABLE OUT two-input AND
+ T_TRIAND2 IO_STD
U2  INV3 $G_DPWR $G_DGND 3 100 5 ; INVerter
+ T_TRIINV IO_STD
U13 NAND3A(2,4) $G_DPWR $G_DGND ; four two-input NAND
+ INA0 IN01 INB0 INB1 INC0 INC1 IND0 IND1
+ ENABLE OUTA OUTB OUTC OUTD
+ T_TRINAND IO_STD

.MODEL T_TRIAND2 UTGATE ; TRI-AND2 Timing Model
+ TPLHMN=15ns TPLHTY=20ns TPLHMX=25ns ...
+ TPZHMN=10ns TPZHTY=15ns TPZHMX=20ns
+ }

Arguments and options

<no. of inputs>
The number of inputs per gate.

<no. of gates>
The number of gates in model.

Comments

In gate arrays the order of the nodes is: all inputs for the first gate, all inputs for the second gate, ..., enable, output for the first gate, output for the second gate, ... In other words, all of the input nodes come first, then the enable, then all of the output nodes. The total number of input nodes is <no. of inputs><no. of gates>+1; the number of output nodes is <no. of gates>. If a tristate gate is connected to a net that has at least one device input using an INLD I/O model, or a device output using an OUTLD I/O model where both parameters are greater than zero, then that net is simulated as a charge storage net.
## Tristate gate types

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameters</th>
<th>Nodes*</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND3</td>
<td>(&lt;no. of inputs&gt;)</td>
<td>in*, en, out</td>
<td>AND gate</td>
</tr>
<tr>
<td>AND3A</td>
<td>(&lt;no. of inputs&gt;,&lt;no. of gates&gt;)</td>
<td>in*, en, out*</td>
<td>AND gate array</td>
</tr>
<tr>
<td>BUF3</td>
<td></td>
<td>in, en, out</td>
<td>Buffer</td>
</tr>
<tr>
<td>BUF3A</td>
<td>(&lt;no. of gates&gt;)</td>
<td>in*, en, out*</td>
<td>Buffer array</td>
</tr>
<tr>
<td>INV3</td>
<td></td>
<td>in, en, out</td>
<td>Inverter</td>
</tr>
<tr>
<td>INV3A</td>
<td>(&lt;no. of gates&gt;)</td>
<td>in*, en, out*</td>
<td>Inverter array</td>
</tr>
<tr>
<td>NAND3</td>
<td>(&lt;no. of inputs&gt;)</td>
<td>in*, en, out</td>
<td>NAND gate</td>
</tr>
<tr>
<td>NAND3A</td>
<td>(&lt;no. of inputs&gt;,&lt;no. of gates&gt;)</td>
<td>in*, en, out*</td>
<td>NAND gate array</td>
</tr>
<tr>
<td>NOR3</td>
<td>(&lt;no. of inputs&gt;)</td>
<td>in*, en, out</td>
<td>NOR gate</td>
</tr>
<tr>
<td>NOR3A</td>
<td>(&lt;no. of inputs&gt;,&lt;no. of gates&gt;)</td>
<td>in*, en, out*</td>
<td>NOR gate array</td>
</tr>
<tr>
<td>NXOR3</td>
<td></td>
<td>in1, in2, en, out</td>
<td>Exclusive NOR gate</td>
</tr>
<tr>
<td>NXOR3A</td>
<td>(&lt;no. of gates&gt;)</td>
<td>in*, en, out*</td>
<td>Excl. NOR gate array</td>
</tr>
<tr>
<td>OR3</td>
<td>(&lt;no. of inputs&gt;)</td>
<td>in*, en, out</td>
<td>OR gate</td>
</tr>
<tr>
<td>OR3A</td>
<td>(&lt;no. of inputs&gt;,&lt;no. of gates&gt;)</td>
<td>in*, en, out*</td>
<td>OR gate array</td>
</tr>
<tr>
<td>XOR3</td>
<td></td>
<td>in1, in2, en, out</td>
<td>Exclusive OR gate</td>
</tr>
<tr>
<td>XOR3A</td>
<td>(&lt;no. of gates&gt;)</td>
<td>in*, en, out*</td>
<td>Excl. OR gate array</td>
</tr>
</tbody>
</table>

* in* and out*—Mean one or more nodes present.
in and out—Refer to only one node.
en—Refers to the output enable node.
# Tristate gate timing model parameters

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPLHMMN</td>
<td>Delay: low to high, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPLHTY</td>
<td>Delay: low to high, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPLHMX</td>
<td>Delay: low to high, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPHLMN</td>
<td>Delay: high to low, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPHLTY</td>
<td>Delay: high to low, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPHLMX</td>
<td>Delay: high to low, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPHZMN</td>
<td>Delay: high to Z, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPHZTY</td>
<td>Delay: high to Z, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPHZMX</td>
<td>Delay: high to Z, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPLZMN</td>
<td>Delay: low to Z, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPLZTY</td>
<td>Delay: low to Z, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPLZMX</td>
<td>Delay: low to Z, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPZLMN</td>
<td>Delay: Z to low, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPZLTY</td>
<td>Delay: Z to low, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPZLMX</td>
<td>Delay: Z to low, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPZHMN</td>
<td>Delay: Z to high, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPZHTY</td>
<td>Delay: Z to high, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPZHMX</td>
<td>Delay: Z to high, max</td>
<td>sec</td>
<td>0</td>
</tr>
</tbody>
</table>

* See .MODEL statement.
Bidirectional transfer gates

The bidirectional transfer gate is a passive device that connects or disconnects two nodes. Bidirectional transfer gates have no parameters.

The state of the gate input controls whether the gate connects the two digital nets. The device type NBTG connects the nodes if the gate is one, and disconnects the nodes if the gate is zero. Device type PBTG connects the nodes if the gate is zero and disconnects the nodes if the gate is one.

The I/O Model DRVH and DRVL parameters are used as a ceiling on the strength of a one or zero, which is passed through a bidirectional transfer gate. If a bidirectional transfer gate is connected to a net which has at least one device input using an INLD I/O model parameter greater than zero, or a device output using an OUTLD I/O model parameter greater than zero, then that net is simulated as a charge storage net.

Device format

U<name> NBTG
+ <digital power node> <digital ground node>
+ <gate node> <channel node 1> <channel node 2>
+ <timing model name> <I/O model name>
+ [MNTYMXDLY = <delay select value>]
+ [IO_LEVEL = <interface subckt select value>]

U<name> PBTG
+ <digital power node> <digital ground node>
+ <gate node> <channel node 1> <channel node 2>
+ <timing model name> <I/O model name>
+ [MNTYMXDLY = <delay select value>]
+ [IO_LEVEL = <interface subckt select value>]

Examples

U4 NBTG $G_DPWR $G_DGND GATE SD1 SD2
+ BTG1 IO_BTG
.MODEL BTG1 UBTG

Model form

.MODEL <timing model name> UBTG

Special behavior when the NBTG or PBTG is connected to an analog device

If a channel node of one of these bidirectional transfer gates is connected to an analog device, then the bidirectional transfer gate is removed during simulation and is replaced with the digital-to-analog subcircuit specified by the bidirectional transfer gate’s I/O model. Because the bidirectional transfer gate is passive and bidirectional, this digital-to-analog subcircuit must model the behavior of the whole bidirectional transfer gate, not just convert its digital levels to analog signals. Use this format to define the digital-to-analog subcircuit:

.SUBCKT <DtoA subckt name> <gate node> <channel node 1> <channel node 2>
+ <digital power node> <digital ground node>
+ params: DRVL=0 DRVH=0 OutLD=0 InLD=0

The contents of the subcircuit must model the behavior of the transfer gate in the analog domain, at least for the channel. If the subcircuit’s gate node is connected to analog devices, then PSpice will simulate the gate node as an analog net. If this behavior is not desired (e.g., the gate will be connected to a clock signal, which will slow simulation if it is an analog signal), then the subcircuit should not have any analog devices connected to the gate node.
The gate node has the same behavior if it is connected to an analog net as other digital device pins: the analog-to-digital subcircuit specified by the I/O model and IO_LEVEL is connected between the analog net and the gate pin of the device.

Examples

The first example is a subcircuit that models the switch with an analog gate connection. In some circuit topologies, this may cause large parts of a circuit to convert to analog if a single net is connected to an analog part. To avoid this, use the _D version of the digital-to-analog converter by setting IO_LEVEL to 3 or 4.

```
.model io_nbtg uio (drvh=200 drvl=200 inld=10pf outld=15pf
+   digpower="DIGIFPWR"TstoreMN=10us
+   inR=10MEGdrvZ =5MEG
+AtoD1="AtoD_HC"AtoD2="AtoD_HC"
+AtoD3="AtoD_HC"AtoD4="AtoD_HC"
+DtoA1="DtoA_NBTG"DtoA2="DtoA_NBTG"
+DtoA3="DtoA_NBTG_D"DtoA4="DtoA_NBTG_D"
.model io_pbtg uio (drvh=200 drvl=200 inld=10pf outld=15pf
+   digpower="DIGIFPWR"TstoreMN=10us
+   inR=10MEGdrvZ =5MEG
+AtoD1="AtoD_HC"AtoD2="AtoD_HC"
+AtoD3="AtoD_HC"AtoD4="AtoD_HC"
+DtoA1="DtoA_PBTG"DtoA2="DtoA_PBTG"
+DtoA3="DtoA_PBTG_D"DtoA4="DtoA_PBTG_D"
.model io_nbtgs uio (drvh=200 drvl=200
+   digpower="DIGIFPWR"TstoreMN=10us
+   inR=10MEGdrvZ =5MEG
+AtoD1="AtoD_HC"AtoD2="AtoD_HC"
+AtoD3="AtoD_HC"AtoD4="AtoD_HC"
+DtoA1="DtoA_NBTG"DtoA2="DtoA_NBTG"
+DtoA3="DtoA_NBTG_D"DtoA4="DtoA_NBTG_D"
.model io_pbtgs uio (drvh=200 drvl=200
+   digpower="DIGIFPWR"TstoreMN=10us
+   inR=10MEGdrvZ =5MEG
+AtoD1="AtoD_HC"AtoD2="AtoD_HC"
+AtoD3="AtoD_HC"AtoD4="AtoD_HC"
+DtoA1="DtoA_PBTG"DtoA2="DtoA_PBTG"
+DtoA3="DtoA_PBTG_D"DtoA4="DtoA_PBTG_D"
.model btg1   ubtg
```

The next two examples are switch models with digital gate inputs. The digital-to-analog conversion of the gate inputs uses an I/O model (HC in this example) that is defined here, not the I/O model of the device driving the gate.

Use these examples in cases where an using analog input would create too many analog switches. Do not use these when the gate is analog, since this would make an analog-to-digital-to-analog conversion, which may cause invalid simulation results. (This is because the analog gate is squared up before being converted to analog again and applied to the “gate” of the switch.)
.subckt DtoA_NBTG gate sd1 sd2 pwr gnd
+params: DRVL=0 DRVH=0 INLD=0 OUTLD=0 VTH=.9 VSAT=1.2
S1 sd1 sd2 gate gnd nbtg_smod
C1 sd1 gnd {.1pf+outld}
C2 sd2 gnd {.1pf+outld}
C3 gate gnd {.1pf+inld}
.model nbtg_smod vswitch
+ (ron={drvl+drvh}/2) roff=1meg von={VSAT} voff={VTH})
.ends

.subckt DtoA_PBTG gate sd1 sd2 pwr gnd
+params: DRVL=0 DRVH=0 INLD=0 OUTLD=0 VTH=-0.9 VSAT=-1.2
S1 sd1 sd2 gate pwr pbtg_smod
C1 sd1 pwr {.1pf+outld}
C2 sd2 pwr {.1pf+outld}
C3 gate gnd {.1pf+inld}
.model pbtg_smod vswitch
+ (ron={drvl+drvh}/2) roff=1meg von={VSAT} voff={VTH})
.ends

.subckt DtoA_NBTG_D gate sd1 sd2 pwr gnd
+params: DRVL={DRVL} DRVH={DRVH} CAPACITANCE={INLD}
S1 sd1 sd2 gate_a gnd nbtg_smod
C1 sd1 gnd {.1pf+outld}
C2 sd2 gnd {.1pf+outld}
.model nbtg_smod vswitch
+ (ron={drvl+drvh}/2) roff=1meg von={VSAT} voff={VTH})
.ends

.subckt DtoA_PBTG_D gate sd1 sd2 pwr gnd
+params: DRVL={DRVL} DRVH={DRVH} CAPACITANCE={INLD}
S1 sd1 sd2 gate_a gnd pbtg_smod
C1 sd1 gnd {.1pf+outld}
C2 sd2 gnd {.1pf+outld}
.model pbtg_smod vswitch
+ (ron={drvl+drvh}/2) roff=1meg von={VSAT} voff={VTH})
.ends
Flip-flops and latches

The simulator supports both edge-triggered and gated flip-flops. Edge-triggered flip-flops change state when the clock changes: on the falling edge for JKFFs, on the rising edge for DFFs. Gated flip-flops are often referred to as latches. The state of gated flip-flops follows the input as long as the clock (gate) is high. The state is frozen when the clock (gate) falls. Multiple flip-flops can be specified in each device. This allows direct modeling of parts which contain more than one flip-flop in a package.

Initialization

By default, at the beginning of each simulation, all flip-flops and latches are initialized to the unknown state (that is, they output an X). Each device remains in the unknown state until explicitly set or cleared by an active-low pulse on either the preset or clear pins, or until a known state is clocked in.

You can override the X start-up state by setting .OPTIONS (analysis options) DIGINITSTATE to either zero or one. If set to zero, all flip-flops and latches in the circuit are cleared. Likewise, if set to one, all such devices are preset. Any other values produce the default (X) start-up state. The DIGINITSTATE option is useful in situations where the initial state of the flip-flop is unimportant to the function of the circuit, such as a toggle flip-flop in a frequency divider.

It is important to note that if the initial state is set to zero or one, the device still outputs an X at the beginning of the simulation if the inputs would normally produce an X on the output. For example, if the initial state is set to one, but the clock is an X at time zero, Q and QBar both go to X when the simulation begins.

X-level handling

The truth-table for each type of flip-flop and latch is given in the sections that follow. However, how the flip-flops treat X levels on the inputs is not depicted in the truth tables because it can depend on the state of the device.

The rule is as follows: if an input is X, and if changing that input between one and zero would cause the output to change, then the output is set to X. In other words, X is only propagated to the output when necessary. For example: if Q = 0 and PresetBar = X, then Q → X; but if Q = 1 and PresetBar = X, then Q → 1.

Timing violations

The flip-flop and latch primitives have model parameters which specify timing constraints such as setup/hold times and minimum pulse-widths. If these model parameter values are greater than zero, the simulator compares measured times on the inputs against the specified value. See Standard gate timing model parameters and Tristate gate timing model parameters.

The simulator reports flip-flop timing violations as digital simulation warning messages in the .out file. These messages can also be viewed using the Windows version of Probe.
**Edge-triggered flip-flops**

The simulator supports four types of edge-triggered flip-flops:

- D-type flip-flop (DFF), which is positive-edge triggered
- J-K flip-flop (JKFF), which is negative-edge triggered
- Dual-edge D flip-flop (DFFDE), which is selectively positive and/or negative edge triggered
- Dual-edge J-K flip-flop (JKFFDE), which is selectively positive and/or negative edge triggered

**Device format**

```
U<name> DFF (<no. of flip-flops>)
  + <digital power node> <digital ground node>
  + <presetbar node> <clearbar node> <clock node>
  + <d node 1> ... <d node n>
  + <q output 1> ... <q output n>
  + <qbar output 1> ... <qbar output n>
  + <timing model name> <I/O model name>
  + [MNTYMXDLY=<delay select value>]
  + [IO_LEVEL=<interface subckt select value>]
```

```
U<name> JKFF (<no. of flip-flops>)
  + <digital power node> <digital ground node>
  + <presetbar node> <clearbar node> <clockbar node>
  + <j node 1> ... <j node n>
  + <k node 1> ... <k node n>
  + <q output 1> ... <q output n>
  + <qbar output 1> ... <qbar output n>
  + <timing model name> <I/O model name>
  + [MNTYMXDLY=<delay select value>]
  + [IO_LEVEL=<interface subckt select value>]
```

```
U<name> DFFDE(<no. of flip-flops>)
  + <digital power node> <digital ground node>
  + <presetbar node> <clearbar node> <clock node>
  + <positive-edge enable node> <negative-edge enable node>
  + <d node 1> ... <d node n>
  + <q output 1> ... <q output n>
  + <qbar output 1> ... <qbar output n>
  + <timing model name> <I/O model name>
  + [MNTYMXDLY=<delay select value>]
  + [IO_LEVEL=<interface subckt select value>]
```

```
U<name> JKFFDE(<no. of flip-flops>)
  + <digital power node> <digital ground node>
  + <presetbar node> <clearbar node> <clock node>
  + <positive-edge enable node> <negative-edge enable node>
  + <j node 1> ... <j node n>
  + <k node 1> ... <k node n>
  + <q output 1> ... <q output n>
  + <qbar output 1> ... <qbar output n>
  + <timing model name> <I/O model name>
  + [MNTYMXDLY=<delay select value>]
  + [IO_LEVEL=<interface subckt select value>]
```

**Timing model format**

```
.MODEL <timing model name> UEFF [model parameters]
```
Examples

U5 JKFF(1) $G_DPWR $G_DGND PREBAR CLRBAR CLKBAR
* one JK flip-flop
  + J K Q QBAR
  + T_JKFF IO_STD
U2 DFF(2) $G_DPWR $G_DGND PREBAR CLRBAR CLK
* two DFF flip-flops
  + D0 D1 Q0 Q1 QBAR0 QBAR1
  + T_DFF IO_STD

.COMMENT T_JKFF UEFF(...) ; JK Timing Model

Comments

Use <no. of flip-flops> to specify the number of flip-flops in the device. The three nodes,
<presetbar node>, <clearbar node> and <clock(bar) node>, are common to all flip-flops in the
device.

The <positive-edge enable node> and <negative-edge enable node> are common to all
flip-flops in the dual-edge flip-flops.
Edge-triggered flip-flop timing model parameters

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>THDCLKMN</td>
<td>Hold: j/k/d after clk/clkb edge, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>THDCLKTY</td>
<td>Hold: j/k/d after clk/clkb edge, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>THDCLKMX</td>
<td>Hold: j/k/d after clk/clkb edge, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPCLKQLHLMN</td>
<td>Delay: clk/clkb edge to q/qb low to hi, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPCLKQLHTY</td>
<td>Delay: clk/clkb edge to q/qb low to hi, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPCLKQLHMX</td>
<td>Delay: clk/clkb edge to q/qb low to hi, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPPCQLHLMN</td>
<td>Delay: preb/clrb to q/qb low to hi, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPPCQLHTY</td>
<td>Delay: preb/clrb to q/qb low to hi, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPPCQLHMX</td>
<td>Delay: preb/clrb to q/qb low to hi, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUDCLKMN</td>
<td>Setup: j/k/d to clk/clkb edge, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUDCLKTY</td>
<td>Setup: j/k/d to clk/clkb edge, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUDCLKMX</td>
<td>Setup: j/k/d to clk/clkb edge, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUPCCLKHMN</td>
<td>Setup: preb/clrb hi to clk/clkb edge, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUPCCLKHTY</td>
<td>Setup: preb/clrb hi to clk/clkb edge, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUPCCLKHMX</td>
<td>Setup: preb/clrb hi to clk/clkb edge, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWPCLMN</td>
<td>Min preb/clrb width low, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWPCLTY</td>
<td>Min preb/clrb width low, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWPCLMX</td>
<td>Min preb/clrb width low, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWCLKLMN</td>
<td>Min clk/clkb width low, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWCLKLTY</td>
<td>Min clk/clkb width low, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWCLKLMX</td>
<td>Min clk/clkb width low, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWCLKHMN</td>
<td>Min clk/clkb width hi, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWCLKHTY</td>
<td>Min clk/clkb width hi, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWCLKHMX</td>
<td>Min clk/clkb width hi, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUCECLKMN</td>
<td>Setup: clock enable to clk edge, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUCECLKTY</td>
<td>Setup: clock enable to clk edge, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUCECLKMX</td>
<td>Setup: clock enable to clk edge, max</td>
<td>sec</td>
<td>0</td>
</tr>
</tbody>
</table>
### Edge-triggered flip-flop truth tables DFF and JKFF

#### D-type flip-flop (DFF) truth table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>CLK</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>↑</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
</tr>
</tbody>
</table>

* Shows an unstable condition.

#### J-K flip-flop (JKFF) truth table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>K</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

* Shows an unstable condition.
Digital devices

Digital primitive summary

Edge-triggered flip-flop truth tables DFFDE and JKFFDE

Dual-edge D flip-flop (DFFDE) truth table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>CLK</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>↑</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
</tr>
<tr>
<td>0</td>
<td>↓</td>
</tr>
<tr>
<td>1</td>
<td>↓</td>
</tr>
</tbody>
</table>

* Shows an unstable condition.

Dual-edge J-K flip-flop (JKFFDE) truth table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>K</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

* Shows an unstable condition.
Gated latch

The simulator supports two types of gated latches: the S-R flip-flop (SRFF) and the D-type latch (DLTCH).

**Device format**

U<name> SRFF (<no. of flip-flops>)
+ <digital power node> <digital ground node>
+ <presetbar node> <clearbar node> <gate node>
+ <s node 1> ... <s node n>
+ <r node 1> ... <r node n>
+ <q output 1> ... <q output n>
+ <qbar output 1> ... <qbar output n>
+ <timing model name> <I/O model name>
+ [MNTYMXDLY=<delay select value>]
+ [IO_LEVEL=<interface subckt select value>]

U<name> DLTCH (<no. of latches>)
+ <digital power node> <digital ground node>
+ <presetbar node> <clearbar node> <gate node>
+ <d node 1> ... <d node n>
+ <q output 1> ... <q output n>
+ <qbar output 1> ... <qbar output n>
+ <timing model name> <I/O model name>
+ [MNTYMXDLY=<delay select value>]
+ [IO_LEVEL=<interface subckt select value>]

**Model form**

.MODEL <timing model name> UGFF [model parameters]

**Examples**

U5 SRFF(4)$G_DPWR $G_DGND PRESET CLEAR GATE
* four S-R latches
+ S0 S1 S2 S3 R0 R1 R2 R3
+ Q0 Q1 Q2 Q3 QB0 QB1 QB2 QB3
+ T_SRFF IO_STD

U2 DLTCH(8) $G_DPWR $G_DGND PRESET CLEAR GATE
* eight D latches
+ D0 D1 D2 D3 D4 D5 D6 D7
+ Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7
+ QB0 QB1 QB2 QB3 QB4 QB5 QB6 QB7
+ T_DLTCH IO_STD

.MODEL T_SRFF UGFF(...) ; SRFF Timing Model

**Comments**

Use <no. of flip-flops> to specify the number of flip-flops in the device. The three nodes, <presetbar node>, <clearbar node>, and <gate node>, are common to all of the flip-flops in the device.

**Gated latch timing model parameters**

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>THDGMN</td>
<td>Hold: s/r/d after gate edge, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>THDGTY</td>
<td>Hold: s/r/d after gate edge, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>THDGMX</td>
<td>Hold: s/r/d after gate edge, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPDQLHMIN</td>
<td>Delay: s/r/d to q/qb low to hi, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPDQLHTY</td>
<td>Delay: s/r/d to q/qb low to hi, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPDQLHMX</td>
<td>Delay: s/r/d to q/qb low to hi, max</td>
<td>sec</td>
<td>0</td>
</tr>
</tbody>
</table>
## Digital primitive summary

<table>
<thead>
<tr>
<th>Model parameters *</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPDQHLMN</td>
<td>Delay: s/r/d to q/qb hi to low, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPDQHLY</td>
<td>Delay: s/r/d to q/qb hi to low, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPDQLMXX</td>
<td>Delay: s/r/d to q/qb hi to low, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPGQLHMN</td>
<td>Delay: gate to q/qb low to hi, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPGQLHTY</td>
<td>Delay: gate to q/qb low to hi, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPGQLMXX</td>
<td>Delay: gate to q/qb low to hi, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPGQQLMN</td>
<td>Delay: gate to q/qb hi to low, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPGQQLHTY</td>
<td>Delay: gate to q/qb hi to low, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPGQQLMXX</td>
<td>Delay: gate to q/qb hi to low, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPPCQLHLMN</td>
<td>Delay: preb/clrb to q/qb low to hi, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPPCQLHTY</td>
<td>Delay: preb/clrb to q/qb low to hi, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPPCQLMXX</td>
<td>Delay: preb/clrb to q/qb low to hi, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPPCQQLMN</td>
<td>Delay: preb/clrb to q/qb hi to low, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPPCQQLHTY</td>
<td>Delay: preb/clrb to q/qb hi to low, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPPCQQLMXX</td>
<td>Delay: preb/clrb to q/qb hi to low, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUDGMN</td>
<td>Setup: s/r/d to gate edge, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUDGTY</td>
<td>Setup: s/r/d to gate edge, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUDGMX</td>
<td>Setup: s/r/d to gate edge, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUPCGHMN</td>
<td>Setup: preb/clrb hi to gate edge, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUPCGHTY</td>
<td>Setup: preb/clrb hi to gate edge, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUPCGHMX</td>
<td>Setup: preb/clrb hi to gate edge, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWPCLMN</td>
<td>Min preb/clrb width low, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWPCLTY</td>
<td>Min preb/clrb width low, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWPCLMX</td>
<td>Min preb/clrb width low, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWGHMN</td>
<td>Min gate width hi, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWGHTY</td>
<td>Min gate width hi, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWGHMX</td>
<td>Min gate width hi, max</td>
<td>sec</td>
<td>0</td>
</tr>
</tbody>
</table>

* See [MODEL (model definition)](model-definition).
Gated latch truth tables

The function tables for the SRFF and DLTCH primitives are given below.

### S-R flip-flop (SRFF) truth table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>S</strong></td>
<td><strong>R</strong></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

* Shows an unstable condition.

### D-type latch (DLTCH) truth table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>D</strong></td>
<td><strong>GATE</strong></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

* Shows an unstable condition.
Pullup and pulldown

The PULLUP and PULLDN primitives function as digital pullup/pulldown resistors. They have no inputs (other than the digital power and ground nodes). Their output is a one level (pullup) or a zero level (pulldown), having a strength determined by the I/O model.

Device format

U<name> <resistor type> (<number of resistors>)
+ <digital power node> <digital ground node>
+ <output node>*
+ [<I/O model name>]
+ [IO_LEVEL=<interface subckt select value>]

Examples

U5 PULLUP(4) $G_DPWR $G_DGND ; four pullup resistors
+ BUS0 BUS1 BUS2 BUS3 R1K
U2 PULLDN(1) $G_DPWR $G_DGND ; one pulldown resistor
+ 15 R500

Arguments and options

<resistor type>

One of the following:

- **PULLUP** pullup resistor array
- **PULLDN** pulldown resistor array

<number of resistors>

Specifies the number of resistors in the array.

Comments

Notice that PULLUP and PULLDN do not have Timing Models, just I/O models.
Delay line

The output of a delay line follows the input after the delay specified in the Timing Model. Any width pulse can propagate through a delay line. This behavior is different from gates, which don’t propagate a pulse when its width is less than the propagation delay.

The delay line device has no parameters, and only one input and one output node.

**Device format**

```
U<name> DLYLINE
+ <digital power node> <digital ground node>
+ <input node> <output node>
+ <timing model name> <I/O model name>
  + [MNTYMXDLY=<delay select value>]
  + [IO_LEVEL=<interface subckt select value>]
```

**Examples**

```
U5 DLYLINE $G_DPWR $G_DGND IN OUT; delay line
+ DLY20NS IO_STD
.MODEL DLY20NS UDLY(  ; delay line Timing Model
+ DLYMN=20ns DLYTY=20ns DLYMX=20ns
+ )
```

**Timing model format**

```
.MMODEL <timing model name> UDLY [model parameters]
```

### Delay line timing model parameters

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLYMN</td>
<td>Delay: min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>DLYTY</td>
<td>Delay: typical</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>DLYMX</td>
<td>Delay: max</td>
<td>sec</td>
<td>0</td>
</tr>
</tbody>
</table>

* See `.MODEL (model definition)`.
Programmable logic array

The programmable logic array is made up of a variable number of inputs, which form columns, and a variable number of outputs, which form rows. Each output (row) is driven by one logic gate. The “program” for the device determines which of the inputs (columns) are connected to each gate. All of the gates in the array are the same type (e.g., AND, OR, NAND, and NOR). Commercially available ICs (PALs, GALs, PEELs, and such) can have buffers, registers, more than one array of gates, and so on, all on the same part. These would normally be combined in a library subcircuit to make the part easier to use.

There are two ways to provide the program data for Programmable Logic Arrays. The normal way is to give the name of a JEDEC format file which contains the program data. This file would normally be produced by a PLD design package, or by using MicroSim PLSyn, which translates logic design information into a program for a specific programmable logic part. The other way to program the logic array is by including the program data, in order, on the device line (using the DATA=... construct).

If one of the PAL or GAL devices are being used in the model library, you will not need to use the Programmable Logic Array primitive directly, nor any of the model information below, since the library contains all of the appropriate modeling information. Using a PLD from the library is just like using any other logic device from the library, except that the simulator needs to know the name of the JEDEC file which contains the program for that part. A TEXT parameter name JEDEC_FILE is used to specify the file name, as shown in the following example:

X1 IN1 IN2 IN3 IN4 IN5 IN6 IN7 IN8 IN9 IN10 IN11 IN12
+ IN13 IN14
+ OUT1 OUT2 OUT3 OUT4
+ PAL14H4
+ TEXT: JEDEC_FILE = "myprog.jed"

This example creates a 14H4 PAL which is programmed by the JEDEC file myprog.jed.

Device format

U<name> <pld type> (<no. of inputs>, <no. of outputs>)
+ <digital power node> <digital ground node>
+ <input_node>* <output_node>*
+ <timing model name> <I/O model name>
+ [FILE=<(file name) text value>]
+ [DATA=<radix flag>$<program data>$]
+ [MNTYMXDLY=<delay select value>]
+ [IO_LEVEL=<interface subckt select value>]

Timing model format

.MODEL <timing model name> UPLD [model parameters]
Examples

UDECODE PLANDC(3, 8) ; 3 inputs, 8 outputs
+ $G_DPWR $G_DGND ; digital power supply and ground
+ IN1 IN2 IN3 ; the inputs
+ OUT0 OUT1 OUT2 OUT3 OUT4 OUT5 OUT6 OUT7 ; the outputs
+ PLD_MDL ; the timing model name
+ IO_STD ; the I/O model name
+ DATA=B$ ; the programming data
* IN1 IN2 IN3
* TF TF TF
+ 01 01 01 ; OUT0
+ 01 01 10 ; OUT1
+ 01 10 01 ; OUT2
+ 01 10 10 ; OUT3
+ 10 01 01 ; OUT4
+ 10 01 10 ; OUT5
+ 10 10 01 ; OUT6
+ 10 10 10 $ ; OUT7

.MODEL PLD_MDL UPLD(...) ; PLD timing model definition

Arguments and options

<pld type>
One of the following:

<table>
<thead>
<tr>
<th>PLD type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLAND</td>
<td>AND array</td>
</tr>
<tr>
<td>PLANDC</td>
<td>AND array using true and complement columns for each input</td>
</tr>
<tr>
<td>PLNAND</td>
<td>NAND array</td>
</tr>
<tr>
<td>PLNANDC</td>
<td>NAND array using true and complement columns for each input</td>
</tr>
<tr>
<td>PLNOR</td>
<td>NOR array</td>
</tr>
<tr>
<td>PLNORC</td>
<td>NOR array using true and complement columns for each input</td>
</tr>
<tr>
<td>PLNXOR</td>
<td>Exclusive NOR array</td>
</tr>
<tr>
<td>PLNXORC</td>
<td>Exclusive NOR array using true and complement columns for each input</td>
</tr>
<tr>
<td>PLOR</td>
<td>OR array</td>
</tr>
<tr>
<td>PLORC</td>
<td>OR array using true and complement columns for each input</td>
</tr>
<tr>
<td>PLXOR</td>
<td>Exclusive OR array</td>
</tr>
<tr>
<td>PLXORC</td>
<td>Exclusive OR array using true and complement columns for each input</td>
</tr>
</tbody>
</table>
<file name text value>

The name of a JEDEC format file which specifies the programming data for the array. The file name can be specified as a text constant (enclosed in double quotes “ ”), or as a text expression (enclosed in vertical bars “|”). If a FILE is specified, any programming data specified by a DATA section is ignored. The mapping of addresses in the JEDEC file to locations in the array is controlled by model parameters specified in the timing model.

<radix flag>

One of the following:

B binary data follows
O octal data follows (most significant bit has the lowest address)
X hexadecimal data follows (most significant bit has lowest address)

<program data>

A string of data values used to program the logic array. The values start at address zero, which programs the array for the connection of the first input pin to the gate which drives the first output. A 0 (zero) specifies that the input is not connected to the gate, and a 1 specifies that the input is connected to the gate. (Initially, all inputs are not connected to any gates.) The next value programs the array for the connection of the complement of the first input to the gate which drives the first output (if this is a programmable gate having true and complement inputs) or, the second input connection to the gate which drives the first output. Each additional 1 or 0 programs the connection of the next input or its complement to the gate which drives the first output, until the connection of all inputs (and their complements) to that gate have been programmed. Data values after that, program the connection of inputs to the gate driving the second output, and so on.

The data values must be enclosed in dollar signs ($), but can be separated by spaces or continuation lines.

Comments

The example defines a 3-to-8 line decoder. The inputs are IN1 (MSB), IN2, and IN3 (LSB). If the inputs are all low, OUT0 is true. If IN1 and IN2 are low and IN3 is high, then OUT1 is true, and so on. The programming data has been typed in as an array, so that it is easier to read. The comments above the columns identify the true and false (complement) inputs, and the comments at the end of the line identify the output pin which is controlled by that gate. (Note, the simulator does not process any of these comments—they just help make the programming data readable.)
Programmable logic array timing model parameters

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPOFFSET</td>
<td>JEDEC file mapping: address of complement of first input and first gate program</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>INSCALE</td>
<td>JEDEC file mapping: amount the JEDEC file address changes for each new input pin</td>
<td>std 1</td>
<td>true/cmp 2</td>
</tr>
<tr>
<td>OFFSET</td>
<td>JEDEC file mapping: address of first input and first gate program</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>OUTSCALE</td>
<td>JEDEC file mapping: amount the JEDEC file address changes for each new output pin (gate)</td>
<td>std &lt;no. of inputs&gt;</td>
<td>true/cmp 2*&lt;no. of inputs&gt;</td>
</tr>
<tr>
<td>TPHLMN</td>
<td>delay: in to out, hi to low, min</td>
<td>sec 0</td>
<td></td>
</tr>
<tr>
<td>TPHLTY</td>
<td>delay: in to out, hi to low, typ</td>
<td>sec 0</td>
<td></td>
</tr>
<tr>
<td>TPHLMX</td>
<td>delay: in to out, hi to low, max</td>
<td>sec 0</td>
<td></td>
</tr>
<tr>
<td>TPLHMM</td>
<td>delay: in to out, low to hi, min</td>
<td>sec 0</td>
<td></td>
</tr>
<tr>
<td>TPLHTY</td>
<td>delay: in to out, low to hi, typ</td>
<td>sec 0</td>
<td></td>
</tr>
<tr>
<td>TPLHMX</td>
<td>delay: in to out, low to hi, max</td>
<td>sec 0</td>
<td></td>
</tr>
</tbody>
</table>

* See .MODEL (model definition).
Read only memory

There are two ways to provide the program data for ROMs. The normal way is to provide the name of an Intel Hex Format file. This file is read before the simulation starts, and the ROM is programmed to contain the data in the file. The other way to program the ROM is to include the program data on the device line (with the DATA=... construct).

The example below defines a 4-bit by 4-bit to 8-bit multiplier ROM.

**Device format**

U<name> ROM( <no. of address pins>, <no. of output pins> )
+ <digital power node> <digital ground node>
+ <enable_node> <address node msb> ... <address node lsb>
+ <output node msb> ... <output node lsb>
+ <timing model name> <I/O model name>
+ [FILE=<file name text value>]
+ [DATA=<radix flag>$<program data>$]
+ [MNTYMXDLY=<delay select value>]
+ [IO_LEVEL=<interface subckt select value>]

**Timing model format**

.MODEL <timing model name> UROM ((<model parameters>*)
Examples

UMULTIPLY ROM(8, 8) ; 8 address bits, 8 outputs
+ $G_DPWR $G_DGND; digital power supply and ground
+ ENABLE ; enable node
+ AIN3 AIN2 AIN1 AINO ; the first 4 bits of address
+ BIN3 BIN2 BIN1 BINO ; the second 4 bits of address
+ OUT7 OUT6 OUT5 OUT4 OUT3 OUT2 OUT1 OUT0 ; the outputs
+ ROM_MDL ; the Timing Model name
+ IO_STD ; the I/O MODEL name
+ DATA=X$ ; the programming data

* B input value:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>A</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>04</td>
<td>05</td>
<td>06</td>
<td>07</td>
<td>08</td>
<td>09</td>
<td>0A</td>
</tr>
<tr>
<td>0B</td>
<td>0C</td>
<td>0D</td>
<td>0E</td>
<td>0F</td>
<td>00</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>04</td>
<td>05</td>
<td>06</td>
<td>07</td>
</tr>
<tr>
<td>+</td>
<td>00</td>
<td>02</td>
<td>04</td>
<td>06</td>
<td>08</td>
<td>0A</td>
<td>0C</td>
<td>0E</td>
<td>10</td>
<td>12</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>A</td>
<td>18</td>
<td>1A</td>
<td>1C</td>
<td>1E</td>
<td>00</td>
<td>02</td>
<td>04</td>
<td>06</td>
<td>08</td>
<td>0A</td>
<td>0C</td>
<td>0E</td>
</tr>
<tr>
<td>16</td>
<td>18</td>
<td>1A</td>
<td>1C</td>
<td>1E</td>
<td>00</td>
<td>02</td>
<td>04</td>
<td>06</td>
<td>08</td>
<td>0A</td>
<td>0C</td>
<td>0E</td>
</tr>
<tr>
<td>+</td>
<td>00</td>
<td>03</td>
<td>06</td>
<td>09</td>
<td>0C</td>
<td>0F</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td>1B</td>
<td>1E</td>
<td>21</td>
</tr>
<tr>
<td>21</td>
<td>24</td>
<td>27</td>
<td>2A</td>
<td>2D</td>
<td>00</td>
<td>04</td>
<td>08</td>
<td>0C</td>
<td>10</td>
<td>14</td>
<td>18</td>
<td>1C</td>
</tr>
<tr>
<td>+</td>
<td>00</td>
<td>04</td>
<td>08</td>
<td>0C</td>
<td>0C</td>
<td>0F</td>
<td>10</td>
<td>14</td>
<td>18</td>
<td>1C</td>
<td>20</td>
<td>24</td>
</tr>
<tr>
<td>2C</td>
<td>30</td>
<td>34</td>
<td>38</td>
<td>3C</td>
<td>10</td>
<td>14</td>
<td>18</td>
<td>1C</td>
<td>20</td>
<td>24</td>
<td>28</td>
<td>32</td>
</tr>
<tr>
<td>+</td>
<td>00</td>
<td>05</td>
<td>0A</td>
<td>0F</td>
<td>14</td>
<td>19</td>
<td>1E</td>
<td>23</td>
<td>28</td>
<td>2D</td>
<td>37</td>
<td>3C</td>
</tr>
<tr>
<td>37</td>
<td>3C</td>
<td>41</td>
<td>46</td>
<td>4B</td>
<td>14</td>
<td>19</td>
<td>1E</td>
<td>23</td>
<td>28</td>
<td>2D</td>
<td>37</td>
<td>3C</td>
</tr>
<tr>
<td>+</td>
<td>00</td>
<td>06</td>
<td>0C</td>
<td>12</td>
<td>18</td>
<td>1E</td>
<td>24</td>
<td>2A</td>
<td>30</td>
<td>36</td>
<td>3C</td>
<td>37</td>
</tr>
<tr>
<td>37</td>
<td>48</td>
<td>4E</td>
<td>54</td>
<td>5A</td>
<td>18</td>
<td>1E</td>
<td>24</td>
<td>2A</td>
<td>30</td>
<td>36</td>
<td>3C</td>
<td>37</td>
</tr>
<tr>
<td>+</td>
<td>00</td>
<td>07</td>
<td>0E</td>
<td>15</td>
<td>1C</td>
<td>23</td>
<td>2A</td>
<td>31</td>
<td>38</td>
<td>3F</td>
<td>46</td>
<td>4D</td>
</tr>
<tr>
<td>4D</td>
<td>54</td>
<td>58</td>
<td>62</td>
<td>69</td>
<td>2A</td>
<td>31</td>
<td>38</td>
<td>3F</td>
<td>46</td>
<td>4D</td>
<td>54</td>
<td>58</td>
</tr>
<tr>
<td>+</td>
<td>00</td>
<td>08</td>
<td>10</td>
<td>18</td>
<td>20</td>
<td>28</td>
<td>30</td>
<td>38</td>
<td>40</td>
<td>48</td>
<td>50</td>
<td>58</td>
</tr>
<tr>
<td>58</td>
<td>60</td>
<td>68</td>
<td>70</td>
<td>78</td>
<td>20</td>
<td>28</td>
<td>30</td>
<td>38</td>
<td>40</td>
<td>48</td>
<td>50</td>
<td>58</td>
</tr>
<tr>
<td>+</td>
<td>00</td>
<td>08</td>
<td>12</td>
<td>1B</td>
<td>24</td>
<td>2D</td>
<td>36</td>
<td>3F</td>
<td>48</td>
<td>51</td>
<td>5A</td>
<td>63</td>
</tr>
<tr>
<td>63</td>
<td>6C</td>
<td>75</td>
<td>7E</td>
<td>87</td>
<td>24</td>
<td>2D</td>
<td>36</td>
<td>3F</td>
<td>48</td>
<td>51</td>
<td>5A</td>
<td>63</td>
</tr>
<tr>
<td>+</td>
<td>00</td>
<td>0A</td>
<td>14</td>
<td>1E</td>
<td>28</td>
<td>32</td>
<td>3C</td>
<td>46</td>
<td>50</td>
<td>5A</td>
<td>64</td>
<td>6E</td>
</tr>
<tr>
<td>6E</td>
<td>78</td>
<td>82</td>
<td>8C</td>
<td>96</td>
<td>28</td>
<td>32</td>
<td>3C</td>
<td>46</td>
<td>50</td>
<td>5A</td>
<td>64</td>
<td>6E</td>
</tr>
<tr>
<td>+</td>
<td>00</td>
<td>0B</td>
<td>16</td>
<td>21</td>
<td>2C</td>
<td>37</td>
<td>42</td>
<td>4D</td>
<td>58</td>
<td>63</td>
<td>6E</td>
<td>79</td>
</tr>
<tr>
<td>79</td>
<td>84</td>
<td>8F</td>
<td>9A</td>
<td>9E</td>
<td>2C</td>
<td>37</td>
<td>42</td>
<td>4D</td>
<td>58</td>
<td>63</td>
<td>6E</td>
<td>79</td>
</tr>
<tr>
<td>+</td>
<td>00</td>
<td>0C</td>
<td>18</td>
<td>24</td>
<td>30</td>
<td>3C</td>
<td>4B</td>
<td>54</td>
<td>60</td>
<td>6C</td>
<td>78</td>
<td>84</td>
</tr>
<tr>
<td>84</td>
<td>90</td>
<td>9C</td>
<td>A8</td>
<td>B4</td>
<td>30</td>
<td>3C</td>
<td>4B</td>
<td>54</td>
<td>60</td>
<td>6C</td>
<td>78</td>
<td>84</td>
</tr>
<tr>
<td>+</td>
<td>00</td>
<td>0D</td>
<td>1A</td>
<td>27</td>
<td>34</td>
<td>41</td>
<td>4E</td>
<td>5B</td>
<td>68</td>
<td>75</td>
<td>82</td>
<td>8F</td>
</tr>
<tr>
<td>8F</td>
<td>9C</td>
<td>A9</td>
<td>B6</td>
<td>C3</td>
<td>34</td>
<td>41</td>
<td>4E</td>
<td>5B</td>
<td>68</td>
<td>75</td>
<td>82</td>
<td>8F</td>
</tr>
<tr>
<td>+</td>
<td>00</td>
<td>0E</td>
<td>1C</td>
<td>2A</td>
<td>3B</td>
<td>46</td>
<td>54</td>
<td>62</td>
<td>70</td>
<td>7E</td>
<td>8C</td>
<td>9A</td>
</tr>
<tr>
<td>9A</td>
<td>8B</td>
<td>B6</td>
<td>C4</td>
<td>D2</td>
<td>3B</td>
<td>46</td>
<td>54</td>
<td>62</td>
<td>70</td>
<td>7E</td>
<td>8C</td>
<td>9A</td>
</tr>
<tr>
<td>+</td>
<td>00</td>
<td>0F</td>
<td>1E</td>
<td>2D</td>
<td>3C</td>
<td>4B</td>
<td>5A</td>
<td>69</td>
<td>78</td>
<td>87</td>
<td>96</td>
<td>A5</td>
</tr>
<tr>
<td>A5</td>
<td>B4</td>
<td>C3</td>
<td>D1</td>
<td>E1</td>
<td>3C</td>
<td>4B</td>
<td>5A</td>
<td>69</td>
<td>78</td>
<td>87</td>
<td>96</td>
<td></td>
</tr>
</tbody>
</table>

.MODEL ROM_MDL UROM(...); ROM Timing Model definition
Arguments and options

<file name text value>
The name of an Intel Hex format file which specifies the programming data for the ROM. The file name can be specified as a text constant (enclosed in double quotes “ ”), or as a text expression (enclosed in vertical bars “|”). If a FILE is specified, any programming data specified by a DATA section is ignored.

<radix flag>
One of the following:

B binary data follows
O octal data follows (most significant bit has lowest address)
X hexadecimal data follows (most significant bit has lowest address)

<program data>
The program data is a string of data values used to program the ROM. The values start at address zero, first output bit. The next bit specifies the next output bit, and so on until all of the output bits for that address have been specified. Then the output values for the next address are given, and so on.

The data values must be enclosed in dollar signs ($ $), but can be separated by spaces or continuation lines.
**Read only memory timing model parameters**

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPADHMN</td>
<td>delay: address to data, low to hi, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPADHTY</td>
<td>delay: address to data, low to hi-Z, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPADHMX</td>
<td>delay: address to data, low to hi, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPADLMN</td>
<td>delay: address to data, hi to low, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPADLTY</td>
<td>delay: address to data, hi to low, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPADLMX</td>
<td>delay: address to data, hi to low, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPEDHMN</td>
<td>delay: enable to data, hi-Z to hi, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPEDHTY</td>
<td>delay: enable to data, hi-Z to hi, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPEDHMX</td>
<td>delay: enable to data, hi-Z to hi, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPEDLMN</td>
<td>Delay: enable to data, hi-Z to low, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPEDLTY</td>
<td>delay: enable to data, hi-Z to low, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPEDLMX</td>
<td>delay: enable to data, hi-Z to low, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPEDHZMN</td>
<td>delay: enable to data, hi to hi-Z, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPEDHZTY</td>
<td>delay: enable to data, hi to hi-Z, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPEDHZMX</td>
<td>delay: enable to data, hi to hi-Z, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPEDLZMN</td>
<td>delay: enable to data, low to hi-Z, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPEDLZTY</td>
<td>delay: enable to data, low to hi-Z, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPEDLZMX</td>
<td>delay: enable to data, low to hi-Z, max</td>
<td>sec</td>
<td>0</td>
</tr>
</tbody>
</table>

*See .MODEL (model definition).*
Random access read-write memory

The RAM is normally initialized using unknown data at all addresses. There are two ways to provide other initialization data for RAMs. The normal way is to give the name of an Intel Hex Format file. This file is read before the simulation starts, and the RAM is initialized to match the data in the file. The other way to initialize the RAM is to include the initialization data on the device line (using the DATA=... construct).

Device format

U<name> RAM(<no. of address bits>, <no. of output bits>)
+ <digital power node> <digital ground node>
+ <read enable node> <write enable node>
+ <address msb node>...<address lsb node>
+ <write-data msb node>...<write-data lsb node>
+ <read-data msb node>...<read-data lsb node>
+ <timing model name> <I/O model name>
+ [MNTYMXDLY=<delay select value>]
+ [IO_LEVEL=<interface subckt select value>]
+ [FILE=<file name text value>]
+ [DATA=<radix flag>$<initialization data>$]

Timing model format

.MODEL <timing model name> URAM (<model parameters>*)

Arguments and options

=file name text value>
The name of an Intel Hex format file which specifies the initialization data for the RAM. The file name can be specified as a text constant (enclosed in double quotes “ “), or as a text expression (enclosed in vertical bars ||). If a FILE is specified, any initialization data specified by a DATA section is ignored.

=radix flag>
One of the following:

B binary data follows
O octal data follows (most significant bit has the lowest address)
X hexadecimal data follows (most significant bit has the lowest address)

=initialization data>
A string of data values used to initialize the RAM. The values start at address zero, first output bit. The next bit specifies the next output bit, and so on until all of the output bits for that address have been specified. Then the output values for the next address are given, and so on.

The data values must be enclosed in dollar signs ($ $), but can be separated by spaces or continuation lines.

The initialization of a RAM using the DATA=... construct is the same as the programming of a ROM. See Read only memory on the ROM primitive for an example.
The RAM has separate read and write sections, using separate data and enable pins, and shared address pins. To write to the RAM, the address and write data signals must be stable for the appropriate setup times, then write enable is raised. It must stay high for at least the minimum time, then fall. Address and data must remain stable while write enable is high, and for the hold time after it falls. Write enable must remain low for at least the minimum time before changing.

To read from the RAM, raise read enable, and the outputs change from Z (high impedance) to the appropriate value after a delay. The address can change while read enable is high, and if it does, the new data is available at the outputs after the delay.

Nothing prevents both the read and write enable from being true at the same time, although most real devices would not allow this. The new value from the write is sent to the read data outputs on the falling edge of write enable.

### Random access memory timing model parameters

<table>
<thead>
<tr>
<th>Model parameters *</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPADHMN</td>
<td>delay: address to read data, low to hi, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPADHTY</td>
<td>delay: address to read data, low to hi, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPADHMX</td>
<td>delay: address to read data, low to hi, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPADLMN</td>
<td>delay: address to read data, hi to low, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPADLTY</td>
<td>delay: address to read data, hi to low, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPADLMX</td>
<td>delay: address to read data, hi to low, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPERDHMN</td>
<td>delay: read enable to read data, hi-Z to hi, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPERDHTY</td>
<td>delay: read enable to read data, hi-Z to hi, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPERDHMX</td>
<td>delay: read enable to read data, hi-Z to hi, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPERDLMN</td>
<td>delay: read enable to read data, hi-Z to low, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPERDLTY</td>
<td>delay: read enable to read data, hi-Z to low, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPERDLMX</td>
<td>delay: read enable to read data, hi-Z to low, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPERDHZMN</td>
<td>delay: read enable to read data, hi to hi-Z, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPERDHZTY</td>
<td>delay: read enable to read data, hi to hi-Z, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPERDHZMX</td>
<td>delay: read enable to read data, hi to hi-Z, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>THAEWTY</td>
<td>min hold time: write enable fall to address change, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>THAEWMX</td>
<td>min hold time: write enable fall to address change, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>THDEWMN</td>
<td>min hold time: write enable fall to data change, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>THDEWTY</td>
<td>min hold time: write enable fall to data change, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>THDEWMX</td>
<td>min hold time: write enable fall to data change, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>THAEWMN</td>
<td>min hold time: write enable fall to address change, min</td>
<td>sec</td>
<td>0</td>
</tr>
</tbody>
</table>
## Digital devices

Digital primitive summary

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPERDLZMN</td>
<td>delay: read enable to read data, low to hi-Z, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPERDLZTY</td>
<td>delay: read enable to read data, low to hi-Z, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPERDLZMX</td>
<td>delay: read enable to read data, low to hi-Z, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUDEEWMN</td>
<td>min setup time: data to write enable rise, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUDEEWTY</td>
<td>min setup time: data to write enable rise, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUDEEWMX</td>
<td>min setup time: data to write enable rise, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUAEWMN</td>
<td>min setup time: address to write enable rise, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUAEWTY</td>
<td>min setup time: address to write enable rise, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSUAEWWMX</td>
<td>min setup time: address to write enable rise, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWEWHMN</td>
<td>min width: enable write hi, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWEWHTY</td>
<td>min width: enable write hi, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWEWHMX</td>
<td>min width: enable write hi, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWEWLMN</td>
<td>min width: enable write low, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWEWLTY</td>
<td>min width: enable write low, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TWEWLMX</td>
<td>min width: enable write low, max</td>
<td>sec</td>
<td>0</td>
</tr>
</tbody>
</table>

* See [MODEL (model definition)](#).
Multi-bit A/D and D/A converter

The simulator provides two primitives to model analog-to-digital converters and digital-to-analog converters: the ADC and the DAC. These two primitives simplify the modeling of these complex mixed-signal devices.
Multi-bit analog-to-digital converter

Device format

\[ U\text{name} \text{ADC(\text{number of bits})} \]
+ \langle\text{digital power node}\rangle \langle\text{digital ground node}\rangle
+ \langle\text{in node}\rangle \langle\text{ref node}\rangle \langle\text{gnd node}\rangle \langle\text{convert node}\rangle
+ \langle\text{status node}\rangle \langle\text{over-range node}\rangle
+ \langle\text{output msb node}\rangle \ldots \langle\text{output lsb node}\rangle
+ \langle\text{timing model name}\rangle \langle\text{I/O model name}\rangle
+ \langle\text{MNTYMXDLY=\langledelay select value\rangle}\rangle
+ \langle\text{IO_LEVEL=\langleinterface subckt select value\rangle}\rangle

Timing model format

\[ .\text{MODEL} \langle\text{timing model name}\rangle \text{UADC [model parameters]} \]

Examples

\[ U5 \text{ADC(4)} \$\_DPWR \$\_DGND ; \text{4-bit ADC} \]
+ Sig Ref 0 Conv Stat OvrRng Out3 Out2 Out1 Out0
+ ADCModel IO\_STD

\[ .\text{MODEL} \text{ADCModel UADC(\ldots)} ; \text{Timing Model} \]

Multi-bit A/D converter timing model parameters

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPCSMN</td>
<td>propagation delay: rising edge of convert to rising edge of status, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPCSTY</td>
<td>propagation delay: rising edge of convert to rising edge of status, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPCSMX</td>
<td>propagation delay: rising edge of convert to rising edge of status, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPDSMN</td>
<td>propagation delay: data valid to falling edge of status, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPDSTY</td>
<td>propagation delay: data valid to falling edge of status, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPDSMX</td>
<td>propagation delay: data valid to falling edge of status, max</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPSDMN</td>
<td>propagation delay: rising edge of status to data valid, min</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPSDTY</td>
<td>propagation delay: rising edge of status to data valid, typ</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TPSDMX</td>
<td>propagation delay: rising edge of status to data valid, max</td>
<td>sec</td>
<td>0</td>
</tr>
</tbody>
</table>

* See \[ .\text{MODEL} \text{\langle model definition\rangle} \].
DATA refers to both the data and over-range signals. The Convert pulse can be any width, including zero. If the propagation delay between the rising edge of the Convert signal and the Status signal (tpsd) is zero, the data and over-range do not go to unknown but directly to the new value. There is a resistive load from <ref node> to <gnd node>, and from <in node> to <gnd node>, of 1/GMIN.

The voltage at <in node> and <ref node> with respect to <gnd node> is sampled starting at the rising edge of the Convert signal, and ending when the Status signal becomes high. This gives a sample aperture time of tpcs plus any rising time for Convert. If, during the sample aperture, the output calculated having the minimum <ref node> voltage and maximum <in node> voltage is different from the output calculated having the maximum <ref node> voltage and minimum <in node> voltage, the appropriate output bits are set to the unknown state and a warning message is printed in the output file.

The output is the binary value of the nearest integer to

\[
\frac{V_{\text{in, gnd}}}{V_{\text{ref, gnd}}} \cdot 2^{n_{\text{bits}}}
\]

If this value is greater than \(2^{n_{\text{bits}}}-1\), then all data bits are 1, and over-range is 1. If this value is less than zero, then all data bits are zero, and over-range is 1.
Multi-bit digital-to-analog converter

Device format

U<name> DAC(<number of bits>)
+ <digital power node> <digital ground node>
+ <out node> <ref node> <gnd node>
+ <input msb node> ... <input lsb node>
+ <timing model name> <I/O model name>
+ [MNTYMXDLY=<delay select value>]
+ [IO_LEVEL=<interface subckt select value>]

Timing model format

.MODEL <timing model name> UDAC [model parameters]

Examples

U7 DAC(4) $G_DPW R $G_DGN D ; 4-bit DAC
+ Sig Ref 0 In3 In2 In1 In0
+ DACModel IO_STD

.MODEL DACModel UDAC(...) ; Timing model

Multi-bit D/A converter timing model parameters

<table>
<thead>
<tr>
<th>Model parameters*</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSWMN</td>
<td>Switching time: change in data to analog out stable, min</td>
<td>sec</td>
<td>10ns</td>
</tr>
<tr>
<td>TSWTY</td>
<td>Switching time: change in data to analog out stable, typ</td>
<td>sec</td>
<td>10ns</td>
</tr>
<tr>
<td>TSWMX</td>
<td>Switching time: change in data to analog out stable, max</td>
<td>sec</td>
<td>10ns</td>
</tr>
</tbody>
</table>

* See .MODEL (model definition).

DAC primitive device timing

The DAC is a zero impedance voltage source from <out node> to <gnd node>. The voltage is

\[ V(\text{ref, gnd}) \cdot \left( \frac{\text{binary value of inputs}}{2^{\text{nbits}}} \right) \]

There is a resistance of 1/GMIN between <ref node> and <gnd node>.
If any inputs are unknown (X), the output voltage is halfway between the output voltage if all the X bits were 1 and the output voltage if all the X bits were 0. When an input bit changes, the output voltage changes linearly to the new value during the switching time.
Behavioral primitives

The simulator offers three primitives to aid in the modeling of complex digital devices: the Logic Expression, Pin-to-Pin Delay, and Constraint Checker primitives. These devices are distinct from other primitives in that they allow data-sheet descriptions to be specified more directly, allowing a one-to-one correspondence using the function diagrams and timing specifications.

The Logic Expression primitive, LOGICEXP, uses free-format logic expressions to describe the functional behavior device.

The Pin-To-Pin Delay primitive, PINDLY, describes propagation delays using sets of rules based on the activity on the device inputs.

The Constraint Checker primitive, CONSTRAINT allows a listing of timing rules such as setup/hold times, and minimum pulse widths. When a violation occurs, the simulator issues a message indicating the time of the violation and its cause.
Logic expression

The LOGICEXP primitive allows combinational logic to be expressed in an equation-like style, using standard logic operators, node names, and temporary variables.

Device format

U<name> LOGICEXP ( <no. of inputs>, <no. of outputs> )
+ <digital power node> <digital ground node>
+ <input node 1> ... <input node n>
+ <output node 1> ... <output node n>
+ <timing model name>
+ <I/O model name>
+ [IO_LEVEL = <value>]
+ [MNTYMXDLY = <value>]
+ LOGIC:
  + <logic assignment>*

Timing model format

.getModel <timing model name> UGATE [model parameters]

Arguments and options

LOGIC:
Marks the beginning of a sequence of one or more <logic assignments>. A <logic assignment> can have one of the two following forms:

<output node> = { <logic expression> }
<temporary value> = { <logic expression> }

<output node>
One of the output node names as it appears in the interface list. Assignments to an <output node> causes the result of the <logic expression> to be scheduled on that output pin. Each <output node> must have exactly one assignment.

<temporary value>
Any target of an assignment which is not specified as one of the nodes attached to the device defines a temporary variable. Once assigned, <temporary values> can be used inside subsequent <logic expressions>. They are provided to reduce the complexity and improve the readability of the model. The rules for node names apply to <temporary value> names

<logic expression>
A C-like, infix-notation expression that returns one of the five digital logic levels. Like all other expressions, <logic expressions> must be surrounded by curly braces { }. They can span one or more lines using the + continuation character in the first column position.

The logic operators are listed below from highest-to-lowest precedence.
Logic Expression Operators

~ unary not
& and
^ exclusive or
| or

The allowed operands are:

- <input nodes>
- Previously assigned <temporary values>
- Previously assigned <output nodes>
- <logic constants>: 0, 1, X, R, F

As in other expressions, parentheses ( ) can be used to group subexpressions. Note that these logic operators can also be used in Probe trace definitions.

Comments

The LOGICEXP primitive uses the same timing model as the standard gate primitives, UGATE.

See Standard gate timing model parameters for the list of UGATE model parameters.
Simulation behavior

When a LOGICEXP primitive is evaluated during a transient analysis, the assignment statements using in it are evaluated in the order they were specified in the netlist. The logic expressions are evaluated using no delay. When the result is assigned to an output node, it is scheduled on that output pin using the appropriate delay specified in the timing model.

Internal feedback loops are not allowed in expressions. That is, an expression cannot reference a value which has yet to be defined. However, external feedback is allowed if the output node also appears on the list of input nodes.

This example models the functionality of the 74181 Arithmetic/Logic Unit. The logic for the entire part is contained in just one primitive. Timing would be handled by the PINDLY and CONSTRAINT primitives. Refer to any major device manufacturer’s data book for a detailed description of the operation of the 74181.

U74181 LOGICEXP( 14, 8 ) DPWR DGND
+ A0BAR A1BAR A2BAR A3BAR B0BAR B1BAR B2BAR B3BAR S0 S1 S2 S3 M CN
+ LF0BAR LF1BAR LF2BAR LF3BAR LAEQUALB LPBAR LGBAR LCN+4
+ DO_GATE IO_STD
+
+ LOGIC:
  *
  * Intermediate terms:
  *
+ I31 = { ~(B3BAR & S3 & A3BAR) | (A3BAR & S2 & ~B3BAR) )
+ I32 = { ~(~B3BAR & S1) | (S0 & B3BAR) | A3BAR ) }
+
+ I21 = { ~(B2BAR & S3 & A2BAR) | (A2BAR & S2 & ~B2BAR) )
+ I22 = { ~(~B2BAR & S1) | (S0 & B2BAR) | A2BAR ) }
+
+ I11 = { ~(B1BAR & S3 & A1BAR) | (A1BAR & S2 & ~B1BAR) )
+ I12 = { ~(~B1BAR & S1) | (S0 & B1BAR) | A1BAR ) }
+
+ I01 = { ~(B0BAR & S3 & A0BAR) | (A0BAR & S2 & ~B0BAR) )
+ I02 = { ~(~B0BAR & S1) | (S0 & B0BAR) | A0BAR ) }
+
+ MBAR = { ~M }
+ P = { I31 & I21 & I11 & I01 }
  *
  * Output Assignments
  *
+ LF3BAR = { (I31 & ~I32) ^
+ ~( (I21 & I11 & I01 & Cn & MBAR) | (I21 & I11 & I02 & MBAR ) |
+ (I21 & I12 & MBAR) | (I22 & MBAR) )}
+
+ LF2BAR = { (I21 & ~I22) ^
+ ~( (I11 & I01 & Cn & MBAR) | (I11 & I02 & MBAR) |
+ (I12 & MBAR) ) }
+
+ LF1BAR = { (I11 & ~I12) ^ ~( (Cn & I01 & MBAR) |
+ (I02 & MBAR) ) }
+
+ LF0BAR = { (I01 & ~I02) ^ ~(MBAR & Cn) }
+
+ LGBAR = { ~( I32 | (I31 & I22) | (I31 & I21 & I12) |
+ (I31 & I22 & I11 & I02) ) }
+
+ LCN+4 = { ~LGBAR | (P & Cn) }
+ LPBAR = { ~P }
+ LAEQUALB = { LF3BAR & LF2BAR & LF1BAR & LF0BAR }
Pin-to-pin delay

The pin-to-pin (PINDLY) primitive is a general mechanism that allows the modeling of complex device timing. It can be thought of as a set of delay-lines (paths) and rules describing how to associate specific amounts of delay using each path.

A PINDLY primitive is used in the output path of a device model, typically at the output pins of a subcircuit definition. A single PINDLY primitive can model the timing and output characteristics of an entire part, including tristate behavior.

PINDLY primitives are expressed and evaluated in a manner similar to the LOGICEXP primitive, except in this case a delay expression is assigned to each output. Whenever an output path undergoes a transition, its delay expression is evaluated to determine the propagation delay which is to be applied to that change.

A delay expression can contain one or more rules that determine which activity on the part’s inputs is responsible for the output change, for example, “is the output changing because the clock changed or the data changed?” This allows device models to be derived directly from data sheets, which typically specify propagation delays based on which input is changing. The PINDLY primitive uses its reference inputs to determine the logic state and recent transitions on nodes which are not in the output path.

Pin-to-pin delay modeling is much simpler compared to earlier methods, in which input-to-output delays had to be distributed among the low-level primitives used to model the device. The latter method can require a great deal of trial and error because manufacturer’s data sheets do not provide a one-to-one association between the logic diagram and the timing specifications.

PINDLY primitives can also contain constraints such as setup/hold, width, and frequency specifications, like those supported by the CONSTRAINT primitive. When used in the PINDLY primitive, these constraints allow the simulator to propagate hazard conditions and report violations in subsequent logic.
Device format

U<name> PINDLY ( <no. of paths>, <no. of enables>, <no. of refs> )
+ <digital-power-node> <digital-ground-node>
+ <input node 1> ... <input node n>
+ [<enable node 1> ... <enable node n>]
+ [<reference node 1> ... <reference node n>]
+ <output node 1> ... <output node n>
+ <I/O model name>
+ [MNTYMXDLY = <delay select value>]
+ [IO_LEVEL = <interface subckt select value>]
+ [BOOLEAN:
  + <boolean assignment>* ]
+ PINDLY:
  + <delay assignment>*
  + [TRISTATE:
    + ENABLE LO | HI <enable node>
    + <delay assignment>* ]
+ [SETUP_HOLD: <setup-hold-specification> ]
+ [WIDTH: <width-specification> ]
+ [FREQ: <frequency-specification> ]
+ [GENERAL: <general-specification> ]

Examples

U1 PINDLY(4,0,3) $G_DPWR $G_DGND
+ IN1 IN2 IN3 IN4
+ REF1 REF2 REF3
+ OUT1 OUT2 OUT3 OUT4
+ IO_MODEL DO_GATE
+ PINDLY:
  + ...

Arguments and options

<no. of paths>

Specifies the number of input-to-output paths represented by the device; the number of inputs must be equal to the number of outputs. A path is defined as an input-to-output association, having the appropriate delay rules started according to the described conditions.

<no. of enables>

Specifies the number of tristate enable nodes used by the primitive. Enable nodes are used in TRISTATE sections. <no. of enables> can be zero.

<no. of refs>

Specifies the number of reference nodes used by the primitive. Reference nodes are used within delay expressions to get state information about signals which are not in the input-to-output paths. <no. of refs> can be zero.

Comments

The example depicts the relationship and purpose of the different pins on the PINDLY primitive.

The PINDLY primitive can be viewed as four buffers, IN1 to OUT1 through IN4 to OUT4, and three reference nodes which are used by the output delay rules. The figure shows how the reference nodes can be used in one or more set of delay rules. In this case, REF1 and REF2 are used by the delay rules for OUT2, and REF3 is used by the delay rules for OUT1 and OUT4. The figure also shows that OUT2 and OUT3 can share the same delay rules. The remainder of the format description describes how to create delay rules.
**BOOLEAN:** Marks the beginning of a section of one or more <boolean assignments>, which define temporary variables that can be used in subsequent <delay expressions>. BOOLEAN sections can appear in any order within the PINDLY primitive. A <boolean assignment> has the following form:

\[ \text{<boolean variable> } = \{ \text{<boolean-expression> } \} \]

<boolean variable> can be any name which follows the node name rules.

<boolean expression> is a C-like, infix-notation expression which returns the boolean value TRUE or FALSE. Like all other expressions, <boolean expressions> must be surrounded by curly braces {...}. They can span one or more lines by using the + continuation character in the first column position. The boolean operators are listed below from highest-to-lowest precedence:

\[
\begin{align*}
\sim & \quad \text{unary not} \\
== & \quad \text{equality} \\
!= & \quad \text{inequality} \\
& \quad \text{and} \\
^\oplus & \quad \text{exclusive or} \\
| & \quad \text{or}
\end{align*}
\]

All boolean operators take the following boolean values as operands:

- Previously assigned <boolean variables>
- Reference functions (defined below)
- Transition functions (defined below)
- <boolean constants>: TRUE, FALSE

In addition, the == and != operators take logic values, such as <input nodes> and <logic constants>. This allows for a check of the values on nodes; for example, CLEAR == 1 returns TRUE if the current level on the node CLEAR is a logic one and FALSE otherwise.
Reference functions

Reference functions are used to detect changes (transitions) on <reference nodes> or <input nodes>. All reference functions return boolean values, and therefore can be used within any <boolean expression>. Following is the list of available reference functions and their arguments:

- \text{CHANGED} \text{<node>}, \text{<delta time>} 
- \text{CHANGED\_LH} \text{<node>}, \text{<delta time>} 
- \text{CHANGED\_HL} \text{<node>}, \text{<delta time>} 

The \text{CHANGED} function returns TRUE if the specified <node> has undergone any state transition within the past <delta time>, prior to the current simulation time; otherwise it returns FALSE.

Similarly, \text{CHANGED\_LH} returns TRUE if <node> has specifically undergone a low-to-high transition within the past <delta time>; FALSE otherwise. Note that \text{CHANGED\_LH} only looks at the most recent (or current) transition. It cannot, for example, determine if 0 \Æ 1 occurred two transitions ago.

Finally, \text{CHANGED\_HL} is similar to \text{CHANGED\_LH}, but checks for high-to-low transitions.

If a <delta time> is specified zero, the reference functions return TRUE if the node has changed at the current simulation time. This allows all of the functionality of a device to be modeled in zero delay so that the total delay through the device can be described using the delay expressions.
Transition functions

Transition functions are used to determine the state change occurring on the changing output, that is, the <output node> for which the <delay expression> is being evaluated. Like reference functions, transition functions return boolean values. However, they differ from reference functions in that transition functions take no arguments, since they implicitly refer to the changing output at the current time. The transition functions are of the general form:

\[ \text{TRN}_{-pn} \]

where \( p \) is the previous state value and \( n \) is the new state value. State values are taken from the set \{ L H Z $ \}. Where appropriate, the $ can be used to signify don’t care, e.g., a TRN_H$ matches a transition from H to ANY state. Rising states automatically map to High, and Falling states automatically map to Low.

As a term in any boolean expression, for example, TRN_LH takes on a TRUE value if the changing output is propagating a change from zero to one.

Following is the complete set of transition functions.

TRN_LH TRN_LZ TRN_L$ TRN_HL TRN_HZ TRN_H$ TRN_ZL TRN_ZH TRN_Z$ TRN_$L TRN_$H TRN_$Z

The TRN_pZ and TRN_Zn functions return true only if it is used within a TRISTATE section, described below. Although open-collector outputs also transition to a high-impedance Z (instead of H), most data books describe propagation times on open-collector outputs as TPLH or TPHL. Therefore, open-collector output devices should use TRN_LH and TRN_HL, and tristate output devices should use TRN_LZ, TRN_HZ, TRN_ZL, and TRN_ZH.

PINDLY: marks the beginning of a section of one or more <delay assignments>, which are used to associate propagation delays using the PINDLY primitive’s outputs. <delay assignments> are of the form:

\[ <\text{output node}>^* = \{ <\text{delay expression}> \} \]

<output node> is one of the output node names as it appears in the interface list. Each <output node> must have exactly one assignment. Several outputs can share the same delay rules by listing them (separated by spaces or commas) on the left-hand side of the <delay expression>.

<delay expression> is an expression which, when evaluated, returns a triplet (min, typ, max) of delay values. Like all other expressions, <delay expressions> must be surrounded by curly braces \{...\}. They can span one or more lines by using the continuation character in the first column position.

The simplest <delay expression> is a single <delay value>, defined as:

\[ \text{DELAY}(\text{min}, \text{typ}, \text{max}) \]

where <min>, <typ>, and <max> are floating point constants or expressions (involving parameters), expressed in seconds. To specify unknown values, use -1. For example, \( \text{DELAY}(20\text{ns},-1,35\text{ns}) \) specifies a minimum time of 20ns, a default (program-computed) value for typical, and a maximum of 35ns. See Treatment of unspecified propagation delays for more information on default delays.
The delay assignment below specifies the propagation delays through output Y to be:
min=2ns, typ=5ns, and max=9ns.

... + PINDLY:
+ Y = { DELAY(2ns, 5ns, 9ns) }
...

To define more complex, rule-based <delay expressions>, use the CASE function, which has the form:

CASE(
<boolean expression>, <delay expression>,; Rule 1
<boolean expression>, <delay expression>,; Rule 2
... ; ...
<delay expression> ; Default delay
)

The arguments to the CASE function are pairs of <boolean expressions> and <delay expressions>, followed by a final default <delay expression>. <boolean expressions> (described above) can contain <boolean values>, reference functions, and transition functions.

When the CASE function is evaluated, each <boolean expression> is evaluated in order of appearance until one produces a TRUE result. When this occurs, the <delay expression> it is paired with the result of the CASE function, and the evaluation of the CASE is ended. If none of the <boolean expressions> return a TRUE result, the value of the final <delay expression> is used. Because it is possible for all <boolean expressions> to evaluate FALSE, the default delay value must be supplied. Note that each argument to the CASE function must be separated by commas.

... + BOOLEAN:
+ CLOCK = { CHANGED_LH( CLK, 0 ) }
+ PINDLY:
+ QA QB QC QD = {
+ CASE ( + CLOCK & TRN_LH, DELAY(-1,13ns,24ns),
+ CLOCK & TRN_HL, DELAY(-1,18ns,27ns),
+ CHANGED_HL( CLRBAR,0), DELAY(-1,20ns,28ns),
+ DELAY(-1,20ns,28ns) ; Default
+ )
+ }

This example describes the delays through a four-bit counter. It shows how rules can be defined to precisely isolate the cause of the output change. In this example, the boolean variable CLOCK is being defined. It is TRUE whenever the reference input CLK changes from low-to-high at the current simulation time. This is only true if the device functionality is modeled in zero delay.

The four outputs QA through QD all share the same delay expression. The CASE is used to specify different delays when the device is counting or clearing. The first two rules define delays when the device is counting (CLK changing low-to-high); the first when the output (QA through QD) is going from low-to-high, the second from high-to-low.

The third rule simply uses the CHANGED_HL function directly to determine whether CLRBAR is changing, and in this case the specification applies to any change (low-to-high or high-to-low) on the output. The default delay applies to all other output transitions which are not covered by the first three rules.
TRISTATE: marks the beginning of a sequence of one or more <delay assignments>. The TRISTATE section differs from the PINDLY section in that the outputs are controlled by the specified enable node.

Immediately following the TRISTATE keyword, an enable node must be specified using its polarity and the ENABLE keyword:

ENABLE HI <enable node>; Specifies active HI enable
ENABLE LO <enable node>; Specifies active LO enable

The specified <enable node> applies to all <output node> assignments in the current section.

Note that <delay expressions> within a TRISTATE section can contain the transition functions pertaining to the Z state, for example TRN_ZL and TRN_HZ.

The following example demonstrates how an enable node can be used to control more than one output. It also shows that some device outputs can use the standard output (PINDLY) while others use the tristate output. (Delay values have been omitted.)
U1 PINDLY(3,1,2) $G_DPWR $G_DGND
+ IN1 IN2 IN3
+ ENA
+ REF1 REF2
+ OUT1 OUT2 OUT3
+ IO_MODEL
+ TRISTATE:
+ ENABLE LO = ENA
+ OUT1 = {
  + CASE(
    + CHANGED(REF1, 0) & TRN_LH, DELAY(...),
    + CHANGED(REF2, 0), DELAY(...),
    + TRN_ZL, DELAY(...),
    + ... +
  )
+ )
+ OUT3 = {
  + CASE(
    + TRN_LZ, DELAY(...),
    + TRN_HZ, DELAY(...),
    + ... +
  )
+ )
+ PINDLY:
+ OUT2 = {
  + CASE(
    + CHANGED(REF1, 0), DELAY(...),
    + ... +
  )
+ )

1 Each CONSTRAINT clause operates independently of all others within a device.

2 By default, for violations involving <input node>, the message tag propagates to the <output node> having positional correspondence.

3 By default, for violations involving <reference node>, the message tag propagates to ALL <output node>s.

4 The default behavior can be overridden by use of one of the following statements, which can appear anywhere within any constraint clause proper:
   AFFECTS (#OUTPUTS) = <output node> { ... }
   AFFECTS_ALL

5 AFFECTS_NONE is always the default for the GENERAL constraint.

SETUP-HOLD: Marks the beginning of a constraint specification. These WIDTH: constructs have the same syntax as those used in the FREQ: CONSTRAINT primitive (see page 3-308).

GENERAL: When a PINDLY primitive is used, the constraint specifications allow the simulator to not only report timing violations, but also to track the effects of the violations in downstream logic. This allows more serious persistent hazards to be reported. This behavior differs from the CONSTRAINT primitive, which only reports timing violations.
PINDLY primitive simulation behavior

A PINDLY primitive is evaluated whenever any of its <input nodes> or <enable nodes> change. The <input node> is positionally associated using its corresponding <output node>. The BOOLEAN statements up to the output assignment are evaluated first, then the appropriate PINDLY or TRISTATE <delay expression> which has been assigned to the changing <output node> is evaluated. The changing input’s state is then applied to the output, using its delay value.

The following PINDLY primitive models the timing behavior of a 74LS160A counter. This example is derived directly from the device model in the model library.

```
ULS160ADLY PINDLY(5,0,4) DPWR DGND
+ RCO QA QB QC QD ; Inputs
+ CLK LOADBAR ENT CLRBAR; Reference nodes
+ RCO_0 QA_0 QB_0 QC_0 QD_0; Outputs
+ IO_LS MNTYMXDLY={MNTYMXDLY} IO_LEVEL={IO_LEVEL}
+
+ BOOLEAN:
+   CLOCK = { CHANGED_LH(CLK,0) }
+   CNTENT = { CHANGED(ENT,0) }
+
+ PINDLY:
+   QA_O QB_O QC_O QD_O = {
+     CASE(
+       CLOCK & TRN_LH, DELAY(-1,13NS,24NS),
+       CLOCK & TRN_HL, DELAY(-1,18NS,27NS),
+       CHANGED_HL(CLRBAR,0), DELAY(-1,20NS,28NS),
+       DELAY(-1,20NS,28NS); Default
+       )
+   )
+
+   RCO = {
+     CASE(
+       CNTENT, DELAY(-1,9NS,14NS),
+       CLOCK & TRN_LH, DELAY(-1,18NS,35NS),
+       CLOCK & TRN_HL, DELAY(-1,18NS,35NS),
+       DELAY(-1,20NS,35NS); Default
+       )
+   )
```
Constraint checker

The CONSTRAINT primitive provides a general constraint checking mechanism to the digital device modeler. It performs setup and hold time checks, pulse width checks, frequency checks, and includes a general mechanism to allow user-defined conditions to be reported.

The CONSTRAINT primitive only reports timing violations. It does not affect propagated or stored logic state or propagation delays.

Timing specifications are usually given at the device (i.e., package pin) level. Thus, the inputs to the constraint description typically are those of the subcircuit description of the device, after any necessary buffering. CONSTRAINT devices can be used in conjunction with any combination of digital primitives, including gates, logic expressions, and pin-to-pin delay primitives.

Device format

U<name> CONSTRAINT ( <no. of inputs> )
+ <digital power node> <digital ground node>
+ <input node 1> ... <input node n>
+ <I/O model name>
+ [ IO_LEVEL = <interface subckt select value> ]
+ [ BOOLEAN: <boolean assignment>* ] ...
+ [ SETUP_HOLD: <setup_hold specification> ] ...
+ [ WIDTH: <width specification> ] ...
+ [ FREQ: <frequency specification> ] ...
+ [ GENERAL: <general specification> ] ...

BOOLEAN: marks the beginning of a section containing one or more <boolean assignments>, of the form:

<boolean variable> = { <boolean expression> }

BOOLEAN sections can appear in any order within the CONSTRAINT primitive.

The syntax of the <boolean expression> is the same as that defined in the PINDLY primitive reference, having the exception that transition functions have no meaning within the CONSTRAINT primitive.

SETUP_HOLD:

Marks the beginning of a setup/hold constraint specification, which has the following format:

  + SETUP_HOLD:
  +  CLOCK <assertion edge> = <input node>
  +  DATA ( <no. of data inputs> ) = <input node j> ... <input node k>
  +  [ SETUPTIME = <time value> ]
  +  [ HOLDTIME = <time value> ]
  +  [ RELEASETIME = <time value> ]
  +  [ WHEN {<boolean expression>} ]
  +  [ MESSAGE = "<additional message text>" ]
  +  [ ERRORLIMIT = <value> ]
  +  [ AFFECTS_ALL | AFFECTS_NONE |
      AFFECTS (#OUTPUTS) = <output-node-list> ]

One or more sections can be specified in any order. Note that AFFECTS clauses are only allowed in PINDLY primitives.

CLOCK defines the node to be used as the reference for setup/hold/release specification. <assertion edge> is one of LH or HL, and specifies which edge of the CLOCK node the setup/hold time is measured against. The CLOCK node must be specified.
DATA defines one or more nodes to be the nodes whose setup/hold time is being measured. At least one DATA node must be specified.

SETUPTIME defines the minimum time that all DATA nodes must be stable prior to the <assertion edge> of the clock. The <time value> must be a nonnegative constant or expression, expressed in seconds. Some devices have different setup time requirements which depend on whether the data is a low or a high at the time of the clock change. In this case, one or both of the following can be used:

\[
\text{SETUPTIME}_\text{LO} = \text{<time value>}
\]
\[
\text{SETUPTIME}_\text{HI} = \text{<time value>}
\]

instead of SETUPTIME, which defines both low- and high-level specifications. If one or both SETUPTIME_xx specifications is zero, the simulator does not perform a setup check for that data level.

HOLDTIME defines the minimum time that all DATA nodes must be stable after the <assertion edge> of the clock. The <time value> must be a nonnegative constant or expression, expressed in seconds. Some devices have different hold time requirements which depend on whether the data is a low or a high at the time of the clock change. In this case, one or both of the following can be used:

\[
\text{HOLDTIME}_\text{LO} = \text{<time value>}
\]
\[
\text{HOLDTIME}_\text{HI} = \text{<time value>}
\]

instead of HOLDTIME, which defines both low- and high-level specifications. If one or both HOLDTIME_xx specifications is zero, the simulator does not perform a hold check for that data level.

RELEASETIME specifications cause the simulator to perform a special-purpose setup check. In a data sheet, release time (also called recovery time) specifications refer to the minimum time a signal (such as CLEAR) can go inactive before the active CLOCK edge. In other words, release times refer to the position of a specific data edge in relation to the clock edge. For this reason, one or both of the following can be used:

\[
\text{RELEASETIME}_\text{LH} = \text{<time value>}
\]
\[
\text{RELEASETIME}_\text{HL} = \text{<time value>}
\]

instead of RELEASETIME, which defines both LH- and HL-edge specifications. The <time value> must be a nonnegative constant or expression, expressed in seconds.

The difference between the release-time checker and the setup-time checker is that simultaneous CLOCK/DATA changes are never allowed in the release-time check. That is, a nonzero hold time is assumed, even though the HOLDTIME is not specified. This feature allows the data sheet values to be specified for release-times directly in a model. For this reason, release times are usually given alone, and not in conjunction with SETUPTIME or HOLDTIME specifications.
Simulation behavior: CLOCK

The sequence of setup/hold/release checks begins when the CLOCK node undergoes the specified LH or HL transition. At that time, the WHEN expression is evaluated. If the result is TRUE, all checks using nonzero specifications are performed for during this clock cycle. If the result is FALSE, then no setup, hold, or release checks are performed. The WHEN expression is used in device models to block the reporting of violations when the device is not listening to the DATA inputs, such as during a clearing function.

The simulator performs setup-time checks when the CLOCK node undergoes an assertion edge. If the HOLDTIME specification is zero, simultaneous CLOCK/DATA transitions are allowed, however the previous value of DATA is still checked for setup-time. If the HOLDTIME is not zero, simultaneous CLOCK/DATA transitions are reported as a HOLDTIME violation.

The simulator performs hold-time checks on any DATA node that changes after the assertion edge on the CLOCK node. If the SETUPTIME is zero, simultaneous CLOCK/DATA changes are allowed, and the next transition on DATA which occurs before the non-asserting clock edge is checked for a hold-time violation.

The simulator performs release-time checks when the CLOCK node undergoes an assertion edge. Simultaneous CLOCK/DATA transitions are not allowed, and is flagged as a violation.

If either the CLOCK or DATA node is unknown (X) at the time of a check, no violation is reported for that node. This reduces the number of unnecessary warning messages: an X being clocked into a device is usually a symptom of another problem which has already been reported.

The sequence ends when the CLOCK node undergoes the other (non-asserting) edge. At this time, any violations which occurred during that clock cycle are reported. (This makes it possible for violations to appear out of time-order in the .out file.)

WIDTH: Marks the beginning of a minimum pulse-width constraint specification, which has the following format:

+ WIDTH:
+   NODE = <input node>
+   [ MIN_HI = <time value> ]
+   [ MIN_LO = <time value> ]
+   [ WHEN {<boolean expression>} ]
+   [ MESSAGE = "<additional message text>" ]
+   [ ERRORLIMIT = <value> ]
+   [ AFFECTS_ALL | AFFECTS_NONE |
+     AFFECTS (#OUTPUTS) = <output-node-list> ]

One or more sections can be specified in any order. Note that AFFECTS clauses are only allowed in the PINDLY primitive.

NODE defines the input node whose pulse width is to be checked.

MIN_HI specifies the minimum time that the <input node> can remain at a high (1) logic level. The <time value> must be a nonnegative constant or expression, expressed in seconds. If not specified, MIN_HI defaults to 0, meaning that any width HI pulse is allowed.

MIN_LO likewise specifies the minimum time that the <input node> can remain at a low (0) logic level. The <time value> must be a nonnegative constant or expression, expressed in seconds. If not specified, MIN_LO defaults to 0, meaning that any width LO pulse is allowed.
At least one instance of MIN_HI or MIN_LO must appear within a WIDTH specification.

FREQ: marks the beginning of a frequency constraint specification, which has the following format:

```
+ FREQ:
  + NODE = <input node>
  + [ MINFREQ = <frequency value> ]
  + [ MAXFREQ = <frequency value>] +
  + [ WHEN { <boolean expression> } ]
  + [ MESSAGE "<additional message text>" ]
  + [ ERRORLIMIT = <value> ]
  + [ AFFECTS_ALL | AFFECTS_NONE |
      AFFECTS (#OUTPUTS) = <output-node-list> ]
```

One or more sections can be specified in any order. Note that AFFECTS clauses are only allowed in the PINDLY primitive.

NODE defines the input node whose frequency is to be checked.

MINFREQ specifies the minimum frequency allowed on <input node>. The <frequency value> must be a nonnegative floating point constant or expression, expressed in hertz.

MAXFREQ specifies the maximum frequency allowed on <input node>. The <frequency value> must be a nonnegative floating point constant or expression, expressed in hertz.

At least one of MINFREQ or MAXFREQ must be specified within a FREQ specification.

**Simulation Behavior: FREQ**

When performing a MINFREQ check, the simulator reports a violation whenever the duration of a period on the <input node> is greater than 1/<frequency value>. Likewise, when performing a MAXFREQ check, it reports a violation whenever any period is less than 1/<frequency value>. To avoid large numbers of violations, the simulator does not report subsequent violations until after a valid cycle occurs.

Note that the use of maximum FREQ specifications provides a slightly different functionality from that achieved by use of minimum pulse-width checks: in the FREQ specification case, the duty-cycle characteristic of the signal is not measured or constrained in any way, whereas the pulse-width check effectively defines the allowable duty-cycle.

Some clocked state-storage device specifications include information about maximum clock frequency, but omit duty-cycle information.

**GENERAL:** Marks the beginning of a general condition test. GENERAL constraints have the following form:

```
+ GENERAL:
  + WHEN { <boolean expression> } +
  + MESSAGE = "<message text>" +
  + [ ERRORLIMIT = <value> ] +
  + [ AFFECTS_ALL | AFFECTS_NONE |
      AFFECTS (#OUTPUTS) = <output-node-list> ]
```

One or more sections can be specified in any order. Note that AFFECTS clauses are only allowed in the PINDLY primitive. The default for the GENERAL constraint is AFFECTS_NONE.

WHEN is used to define a boolean expression, which can describe arbitrary signal relationships that represent the error or condition of interest.

MESSAGE defines the message to be reported by the simulation whenever the WHEN expression evaluates TRUE. The <message text> must be a text constant (enclosed by double quotes “ ”) or a text expression.

The <boolean expression> is evaluated whenever the CONSTRAINT primitive is evaluated, that is, whenever any of its inputs undergo a transition. If the result is TRUE, the simulator produces a header containing the time of the occurrence, followed by the <message text>.

General notes

Any or all of the constraint specifications (SETUP_HOLD, WIDTH, FREQ, GENERAL) can appear, in any order, within a CONSTRAINT primitive. Further, more than one constraints of the same type can appear (such as two WIDTH specifications). Each of the constraint specifications is evaluated whenever any inputs to the CONSTRAINT primitive instance change.

All constraint specifications can optionally include a WHEN statement, which is interpreted as “only perform the check when result of <boolean expression> == TRUE.” The WHEN statement is required in the GENERAL constraint.

Each constraint type (SETUP_HOLD, WIDTH, FREQ, and GENERAL) has an associated built-in message. In addition, each instance can include a MESSAGE specification, which takes a text constant (enclosed in double quotes “ ”) or text expression. The <additional message text> is appended to the end of the internally-generated, type-specific message which is output whenever a violation occurs. The MESSAGE clause is required for the GENERAL constraint device.

All of the constraint specifications can accept an optional ERRORLIMIT specification. The <value> must be a nonnegative constant or expression. The default <value> is obtained from the value of the DIGERRDEFAULT (set using the .OPTIONS command), which defaults to 20. A value of zero is interpreted as infinity, i.e., no limit. When more than <value> violations of the associated constraint have occurred, no further message output is generated for that constraint checker; other checkers within the CONSTRAINT primitive that have not exceeded their own ERRORLIMITs continue to operate.

During simulation, if the total number of digital violations reported exceeds the value given by DIGERRLIMIT (set using the .OPTIONS (analysis options) command), then the simulation is halted. DIGERRLIMIT defaults to infinity.
This CONSTRAINT primitive example below was derived from the 74LS160A device in the model library. It demonstrates how all of the timing checks can be performed by a single primitive.

```
ULS160ACON CONSTRAINT(10) DPWR DGND
+ CLK ENP ENT CLRBAR LOADBAR A B C D EN
+ IO_LS
+ FREQ:
+ NODE = CLK
+ MAXFREQ = 25MEG
+ WIDTH:
+ NODE = CLK
+ MIN_LO = 25NS
+ MIN_HI = 25NS
+ WIDTH:
+ NODE = CLRBAR
+ MIN_LO = 20NS
+ SETUP_HOLD:
+ DATA(1) = LOADBAR
+ CLOCK LH = CLK
+ SETUPTIME = 20NS
+ HOLDTIME = 3NS
+ WHEN = { CLRBAR!='0 } + SETUP_HOLD:
+ DATA(2) = ENP ENT
+ CLOCK LH = CLK
+ SETUPTIME = 20NS
+ HOLDTIME = 3NS
+ WHEN = { CLRBAR!='0 & (LOADBAR!='0 ^ CHANGED(LOADBAR,0))
+ & CHANGED(EN,20NS) } + SETUP_HOLD:
+ DATA(4) = A B C D
+ CLOCK LH = CLK
+ SETUPTIME = 20NS
+ HOLDTIME = 3NS
+ WHEN = { CLRBAR!='0 & (LOADBAR!='1 ^ CHANGED(LOADBAR,0)) } + SETUP_HOLD:
+ DATA(1) = CLRBAR
+ CLOCK LH = CLK
+ RELEASETIME_LH = 25NS
```
Stimulus devices

Stimulus devices apply digital waveforms to a node. Their purpose is to provide the input to a digital circuit or a digital portion of a mixed circuit. They play the same role in the digital simulator that the independent voltage and current sources (V and I devices) do in the analog simulator.

There are two types of stimulus devices: the stimulus generator (STIM), which uses a simple command to generate a wide variety of waveforms; and the file stimulus (FSTIM), which obtains the waveforms from an external file.

Unlike digital primitives, stimulus devices do not have a Timing Model. This is similar to the analog V and I devices: the timing characteristics are described by the device itself, not in a separate model.
Stimulus generator

Device format

U<name> STIM(<width>, <format array>)
+ <digital power node> <digital ground node>
+ <node>*
+ <I/O model name>
+ [STIMULUS=<stimulus name>]
+ [IO_LEVEL=<interface subckt select value>]
+ [TIMESTEP=<stepsize>]
+ <command>*

Arguments and options

<width>
Specifies the number of signals (nodes) output by the stimulus generator.

<format array>
Specifies the format of <value>s used in defining the stimulus. <format array> is a sequence of digits which specifies the number of signals (nodes) that the corresponding digit in a <value> represents. Each digit of <value> is assumed to be in base 2<sub>m> where <m> is the corresponding digit in <format array>. Each <value> must have the same number of digits as <format array>. The sum of the digits in <format array> must be <width>, and each digit must be either a 1, 3, or 4 (that is, binary, octal, or hexadecimal).

<digital power node> <digital ground node>
These nodes are used by the interface devices which connect analog nodes to digital nodes or vice versa. Refer to your PSpice user’s guide for more information.

<node>*
One or more node names which are output by the stimulus generator. The number of nodes specified must be the same as <width>.

<I/O model name>
The name of an I/O model, which describes the driving characteristics of the stimulus generator. I/O models also contain the names of up to four DtoA interface subcircuits, which are automatically called by the simulator to handle interface nodes. In most cases, the I/O model named IO_STM can be used from the “dig_io.lib” library file. Refer to your PSpice user’s guide for a more detailed description of I/O models.

STIMULUS
An optional parameter for referencing a stimulus definition.

IO_LEVEL
An optional device parameter which selects one of the four DtoA interface subcircuits from the I/O model. The simulator calls the selected subcircuit automatically in the event a <node> connects to an analog device. If not specified, IO_LEVEL defaults to 0. Valid values are:

0 = the current value of .OPTIONS DIGIOVLVL (default=1)
1 = DtoA1
2 = DtoA2
3 = DtoA3
4 = DtoA4

Refer to your PSpice user’s guide for more information.
TIMESTEP
Number of seconds per clock cycle, or step. Transition times that are specified in clock cycles (using the C suffix) are multiplied by this amount to determine the actual time of the transition. (See <time> below.) If TIMESTEP is not specified, the default is zero seconds. TIMESTEP has no effect on <time> values which are specified in seconds (using the S suffix).

<command>*
A description of the stimuli to be generated, using one or more of the following.

<time> <value>
LABEL=<label name>
<time> GOTO <label name> <n> TIMES
<time> GOTO <label name> UNTIL GT <value>
<time> GOTO <label name> UNTIL GE <value>
<time> GOTO <label name> UNTIL LT <value>
<time> GOTO <label name> UNTIL LE <value>
<time> INCR BY <value>
<time> DECR BY <value>
REPEAT FOREVER
REPEAT <n> TIMES
ENDREPEAT
FILE=<file name>

<time>
Specifies the time for the new <value>, GOTO, or INCR/DECR command to occur.

**Time units**

Time values can be stated in seconds or in clock cycles (see TIMESTEP above). To specify a time value in clock cycles, use the C suffix. Otherwise, the units default to seconds.

**Absolute/relative times**

Times can be absolute, such as 45ns or 10c, or relative to the previous time. To specify a relative time, prefix the time using a “+” such as +5ns or +2c.

<value> is the value for each node (0, 1, R, F, X, or Z). <value> is interpreted using the <format array>.

<label name> is the name used in GOTO statements. GOTO <label name> jumps to the next non-label statement after the <LABEL = <label name>> statement.

<n> is the number of times to repeat a GOTO loop. Use a -1 to specify forever.

Keep the following in mind when using the stimulus command:

Transitions using absolute times within a GOTO loop are converted to relative times based on the time of the previous command and the current step size.

- GOTO <label name> must specify a label that has been defined in a previous LABEL=<label name> statement.
- Times must be in strictly ascending order, except that the transition after a GOTO can be at the same time as the GOTO.
A simpler syntax for constructing counted loops in digital stimulus is to use the REPEAT/ENDREPEAT construct. Specify the count value, for example:

```
REPEAT 3 TIMES
+ 5ns 0
+ 5ns 1
ENDREPEAT
```

For an infinite loop, use REPEAT FOREVER (equivalent to REPEAT -1 TIMES). All times within REPEAT loops are interpreted as relative to the start of the loop.

Transition (i.e., time-value pairs) information can be placed in a FILE and accessed one or more times from the STIM device by using the FILE= statement. The syntax for the file contents is identical to what can appear directly in the body of the STIM device <command> section.

**Stimulus generator examples**

**One**  The first example creates a simple reset signal, which could be used to set or clear a flip-flop at the beginning of a simulation. The node, named Reset, is set to a level zero at time zero nanoseconds, and to a Z (high impedance) at 20 ns.

```
UReset STIM(1,1) $G_DPWR $G_DGND
+ Reset
+ IO_STM
+ 0s 0
+ 20ns Z
```

This is useful when the Reset node is being driven by another device which does not reset the flip-flop at time zero. By using the library I/O model named IO_STM, the stimulus generator drives with a high strength, and thus overpowers the other output. By outputting a Z for the duration of the simulation, the stimulus generator cannot affect the node.

**Two**  The second example is a simple example of a clock stimulus which pulses every 5 nanoseconds. It has one output node, OUT1, and the format is represented in binary notation. This example specifies the time as relative to the previous step. IO_STM is an I/O model for stimulus devices and is available in the dig.io.lib library file which comes with the digital simulation feature.

```
UEx2 STIM(1,1) $G_DPWR $G_DGND Out1 IO_STM
+ 0s 0; At time=0 initialize Out1; to zero.
+ REPEAT FOREVER; repeats loop indefinitely
+ +5ns 1 ; 5ns later Out1 is set to 1
+ +5ns 0 ; 5ns later Out1 is set to 0
+ ENDREPEAT
```
Three The third example illustrates the use of the timestep; a cycle is equal to one nanosecond:

```
UEx3 STIM( 2, 11 ) $G_DPWR $G_DGND 1 2
+ IO_STM TIMESTEP=1ns
+ 0c 00 ; At time=0ns, both nodes are set to 0.
+ REPEAT 4 TIMES ; What's in the loop is repeated 4 times
+ +1c 01 ; 1ns later node 1 is set to 0 and node 2 is set to 1.
+ +2c 11 ; 2ns later both nodes set to 1.
+ ENDREPEAT
```

Four The fourth example has four output nodes. The values of the nodes at each transition are in hexadecimal notation. This is because the <format array> is set to 4, meaning <value> is one digit representing the value of four nodes. Both the absolute and relative timing methods are used, but, at the start of execution, the simulation converts all absolute values to relative values based on the time of the command and the current step size. The timestep is equal to one nanosecond, setting the cycle to one nanosecond:

```
UEx4 STIM( 4, 4 ) $G_DPWR $G_DGND IN1 IN2 IN3 IN4
+ IO_STM TIMESTEP=1ns
+ 0s 0 ; At time=0 seconds, all nodes are set to 0.
+ LABEL=STARTLOOP
+ 10C 1 ; At time=10NS, IN1, IN2, & IN3 are set to 0 and IN4 is set to 1.
+ +5NS 0 ; 5NS later, all nodes are set to 0.
+ 20NS A ; At time=20NS, nodes IN1 & IN3 are set to 1 and nodes IN2 & IN4 are to 0.
+ +5NS 0 ; 5NS later, all nodes are set to 0.
+ 30C GOTO STARTLOOP 1 TIMES ; At time=30NS, execute the first statement of the loop without a further delay. “1 TIMES” causes the logic to loop 1 time, actually executing the loop twice.
+ +10C 1 ; After the logic falls through the loop the second time and then waiting 10 additional cycles (or 10 nanoseconds), IN1, IN2, & IN3 are set to 0 and IN4 is set to 1.
```
Example four produces the following transitions. Note how all of the time values are calculated relative to the previous step:

<table>
<thead>
<tr>
<th>TIME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00E+00</td>
<td>0000</td>
</tr>
<tr>
<td>1.00E-08</td>
<td>0001 ; STARTLOOP</td>
</tr>
<tr>
<td>1.50E-08</td>
<td>0000</td>
</tr>
<tr>
<td>2.00E-08</td>
<td>1010 ; 1010 in hex=A</td>
</tr>
<tr>
<td>2.50E-08</td>
<td>0000</td>
</tr>
<tr>
<td>3.00E-08</td>
<td>0001 ; The GOTO STARTLOOP 1 TIMES causes the first statement after the STARTLOOP label to be executed immediately.</td>
</tr>
<tr>
<td>3.50E-08</td>
<td>0000</td>
</tr>
<tr>
<td>4.00E-08</td>
<td>1010</td>
</tr>
<tr>
<td>4.50E-08</td>
<td>0000 ; At time 5.00E-08 we checked the GOTO STARTLOOP 1 TIMES statement, but did not execute it since it was already completed one time.</td>
</tr>
<tr>
<td>6.00E-08</td>
<td>0001 ; At 10C=1ns * 10=10ns later we execute the last statement.</td>
</tr>
</tbody>
</table>

![Diagram of transitions](attachment:image1)

**Five** The fifth example illustrates the use of the INCR BY command used to increment the value of the 16 bit bus:

```
UEx5 STIM ( 16, 4444 ) $G_DPWR $G_DGND
+ 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
+ IO_STM TIMESTEP = 10ns
+ 0s 0000 ; At time=0 seconds, all nodes are set to 0.
+ LABEL=STARTLOOP
+ 10c INCR BY 0001 ; At 100ns, increment bus by 1.
+ 20c GOTO STARTLOOP UNTIL GE 000A ; If the bus value is less than 10, branch back to STARTLOOP and execute the line following the label without a further delay.
```

![Diagram of bus transitions](attachment:image2)
Six

The sixth example has seven output nodes: 1, 2, 3, 4, 5, 6, and 7. The <format array> specifies the notation (1=binary, 3=octal, or 4=hex) used to define the output of those seven nodes. The first two output signals are defined in binary, the next four are in hexadecimal, and the last one is in binary.

In this example, at time equal to one nanosecond, the value of 0070 creates the bit pattern 0001110 on the output nodes. The first two zeros correspond to outputs one and two, the 0111 (7 in hex) corresponds to output signals 3 through 6, and the last zero is the value of output signal 7.

```
UEx6 STIM( 7, 1141 ) $G_DPWR $G_DGND 1 2 3 4 5 6 7 IO_STM
+ 0ns 0000; At time=0ns, all nodes are set to 0.
+ REPEAT 4 TIMES; Repeats what's in loop 4 times.
+ +1ns 0070; At time=1ns, nodes 1, 2, & 3 are set to 0,
+ nodes 4, 5, & 6 are set to 1,
+ and node 7 is set to 0.
+ +2ns 11F1; At time=2ns, all nodes are set to 1.
+ ENDREPEAT
```
File stimulus

The file stimulus device, FSTIM, allows the digital stimuli to be obtained from a file. This is often useful if the number of stimuli is very large, or if the inputs to one simulation come from the output of another simulation (or even from another simulator). To make the discussion of the FSTIM device more meaningful, the stimulus file format is discussed first.

Stimulus file format

The stimulus file has a simple format which allows outputs from other simulators, or the simulation output file, to be used with little modification. The file consists of two sections: the header, which contains a list of signal names, and the transitions, which is one or more lines containing the transition time and columns of values. The header and transitions must be separated by at least one blank line. Below is a simple example of the stimulus file format.

* Header, containing signal names (standard comments are allowed)
Clock, Reset, In1, In2; four signal names

* Beginning of the transitions - note the blank line
0 0000 ; values are in binary
10ns 1100
20ns 0101
30ns 1110
40ns 0111

Header format

[TIMESCALE=<value>]
<signame 1>...<signame n>...
OCT(<signame bit 3> ... <signame lsb>) ...
HEX(<signame bit 4> ... <signame lsb>) ...

The header consists of the list of signal names and an optional TIMESCALE value. The signal names can be separated by commas, spaces, or tabs. The list can span several lines, but must not include the + continuation character. The signal names listed correspond to the columns of values in the order that they are listed. Up to 255 signals can be listed in the header, however a maximum of 300 characters are allowed per line.

The OCT and HEX radix functions allows three or four signals to be grouped, respectively, into a single octal or hexadecimal digit in the columns of values. Note that exactly three signals must be included inside the parentheses in the OCT function, and that exactly four signals must be included in the HEX function. Signal names listed without the radix functions default to binary values.

The following example shows the use of the HEX radix function.

Clock Reset In1 In2
HEX(Addr7 Addr6 Addr5 Addr4) HEX(Addr3 Addr2 Addr1 Addr0)
ReadWrite

0 0000 00 0 ; spaces can be used to group values
10ns 1100 4E 0
20ns 0101 4E 1
30ns 1110 4E 1
40ns 0111 FF 0
In this example, there are four binary signals, followed by two occurrences of the HEX radix function, followed by a single binary signal. In the list of transitions following the header, there are seven values which correspond, in order, to the list of signals.

The optional TIMESCALE assignment is used to scale the time values in the transitions. The TIMESCALE assignment must be on a separate line. If unspecified, TIMESCALE defaults to 1.0. See <time> below for more information on the use of TIMESCALE.

**Transition format**

<time> <value>*  Following the first blank line after the header, the simulator looks for one or more lines containing transitions. Transitions consist of a time value, followed by one or more values corresponding to the signal names in the header. The <time> and list of <values> must be separated by at least one space or tab.

<time>  Transition times are always stated in seconds. Times can be absolute, such as 45ns, 1.2e-8, or 10; or relative to the previous time. To specify relative time, prefix the time using a +, such as +5ns or +1e-9.

Time values are always scaled by the value of TIMESCALE. This is useful if the time values in the file are expressed as whole numbers, but the actual units are, for example, 10ns. An example showing the use of TIMESCALE is given below.

<value>*  Each value corresponds to a single binary signal (the default) or the entire group of signals inside the OCT or HEX radix functions. The number of values listed must equal the total number of binary signals and radix functions which are specified in the header. Valid <values> are:

<table>
<thead>
<tr>
<th></th>
<th>Binary</th>
<th>OCT</th>
<th>HEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic/Numeric</td>
<td>0,1</td>
<td>0-7</td>
<td>0-F</td>
</tr>
<tr>
<td>Unknown</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Hi-impedance</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
</tr>
<tr>
<td>Rising</td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Falling</td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>
When the <value> in a HEX or OCT column is a number, the simulator converts the number to binary and assigns the appropriate logic value of each bit (either zero or one) to the signals inside the radix function. The bits are assigned msb to lsb. When the <value> is X, Z, R, or F, all signals in the radix function take on that value. Note that there can be no falling value in a HEX column because F is used as a numeric value.

The following example shows the use of TIMESCALE and relative <time> values.

```
TIMESCALE=10ns ; must appear on separate line
Clock, Reset, In1, In2
HEX(Addr7 Addr6 Addr5 Addr4) HEX(Addr3 Addr2 Addr1 Addr0)
ReadWrite
  0 0000 00 0
  1 110R 4E 0 ; transition occurs at 10ns
  2 0101 4E 1
  + 3 1111 4E 1 ; transition occurs at 50ns
  7 011F C3 0 ; transition occurs at 70ns
  8 11X0 C3 1
```

### File stimulus device

The file stimulus device, FSTIM, is used to access one or more signals inside a stimulus file. More than one FSTIM device can access the same file. An FSTIM device can even refer to the same signal as another FSTIM device. Any number of stimulus files can be used during a simulation.

#### Device format

```
U<name> FSTIM(<# outputs>)
  + <digital power node> <digital ground node>
  + <node>*
  + <I/O model name>
  + FILE=<stimulus file name>
  + [IO_LEVEL=<interface subckt select value>]
  + [SIGNAMES=<stimulus file signal name>*]
```

#### Examples

```
U1 FSTIM(1) $G_DPWR $G_DGND
  + IN1 IO_STM FILE=DIG1.STM
U2 FSTIM(4) $G_DPWR $G_DGND
  + ADDR3 ADDR2 ADDR1 ADDRO
  + IO_STM
  + FILE = DIG_2.STM
  + SIGNAMES = AD3 AD2 AD1 AD0
U3 FSTIM(4) $G_DPWR $G_DGND
  + CLK PRE J K
  + IO_STM
  + FILE = FLIPFLOP.STM
  + SIGNAMES = CLOCK PRESET
```
Arguments and options

<# outputs> Specifies the number of nodes driven by this device.

<digital power node> <digital ground node>
These nodes are used by the interface devices which connect analog nodes to digital nodes or vice versa. Refer to your PSpice user’s guide for more information.

<node>*
One or more node names which are output by the file stimulus. The number of nodes specified must be the same as <# outputs>.

<I/O model name>
The name of an I/O model, which describes the driving characteristics of the stimulus device. I/O models also contain the names of up to four DtoA interface subcircuits, which are automatically called by the simulator to handle interface nodes. In most cases, the I/O model named IO_STM can be used from the library dig_io.lib. Refer to your PSpice user’s guide for a more detailed description of I/O models.

FILE
The name of the stimulus file to be accessed by this device. The <stimulus file name> can be specified as a quoted string or as a text expression; see .TEXT (text parameter). Note that the FILE device parameter is not optional.

IO_LEVEL
An optional device parameter which selects one of the four AtoD or DtoA interface subcircuits from the device’s I/O model. The simulator calls the selected subcircuit automatically in the event a node connecting to the primitive also connects to an analog device. If not specified, IO_LEVEL defaults to 0. Valid values are:

0 = the current value of .OPTIONS DIGIOLVL (default=1)
1 = AtoD1/DtoA1
2 = AtoD2/DtoA2
3 = AtoD3/DtoA3
4 = AtoD4/DtoA4

Refer to your PSpice user’s guide for more information.

SIGNAMES
Used to specify the names of the signals inside the stimulus file which are to be referenced by the FSTIM device. The signal names correspond, in order, to the <nodes> connected to the device. If any or all SIGNAMES are unspecified, The simulator looks in the stimulus file for the names of the <nodes> given. Because the number of signal names can vary, the SIGNAMES parameter must be specified last. SIGNAMES can be a list of names or text expressions (see .TEXT), or a mixture of the two.
The first example references a file named `dig1.stm`. This file must have a signal named IN1. The second example references `dig2.stm`. This file would have to have signals named AD3 through AD0. These are mapped, in order, to the nodes ADDR3 through ADDR0, which are driven by this device.

In the third example, the FSTIM device references the file `flipflop.stm`. The contents of `flipflop.stm` are shown below:

```
J K PRESET CLEAR CLOCK
0 0 0 010
10ns 0 0 111
```

In this example, the first two nodes, CLK and PRE, reference the signals named CLOCK and PRESET in the stimulus file. The last two nodes, J and K, directly reference the signals named J and K in the file, and therefore do not need to be listed in SIGNAMES. Note that the order of the SIGNAMES on the FSTIM device does not need to match the order of the names listed in the header of the stimulus file. It is not required that every signal in the file be referenced by an FSTIM device. In the example above, the signal named CLEAR is not referenced. One, several, or all signals in a stimulus file can be referenced by one or more FSTIM devices.
Input/output model

Each digital device in the circuit must reference an I/O model. The I/O model describes the device’s loading and driving characteristics. It also contains the names of up to four AtoD and DtoA subcircuits that the simulator calls to handle interface nodes.

I/O models are common to device families. For example, of the digital devices in the model library, there are only four I/O Models for the entire 74LS family: IO_LS, for standard inputs and outputs; IO_LS_OC, for standard inputs and open-collector outputs; IO_LS_ST, for schmitt trigger inputs and standard outputs; and IO_LS_OC_ST, for schmitt trigger inputs and open-collector outputs. This is in contrast to timing models, which are unique to each device in the library.

Model form

```
.MODEL <I/O model name> UIO [model parameters]
```

### Input/output model parameters

<table>
<thead>
<tr>
<th>Model Parameter</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>AtoD1</td>
<td>Name of level 1 AtoD interface subcircuit</td>
<td></td>
<td>AtoDDefault</td>
</tr>
<tr>
<td>AtoD2</td>
<td>Name of level 2 AtoD interface subcircuit</td>
<td></td>
<td>AtoDDefault</td>
</tr>
<tr>
<td>AtoD3</td>
<td>Name of level 3 AtoD interface subcircuit</td>
<td></td>
<td>AtoDDefault</td>
</tr>
<tr>
<td>AtoD4</td>
<td>Name of level 4 AtoD interface subcircuit</td>
<td></td>
<td>AtoDDefault</td>
</tr>
<tr>
<td>DIGPOWER</td>
<td>Name of power supply subcircuit</td>
<td></td>
<td>DIGIFPWR</td>
</tr>
<tr>
<td>DRVH</td>
<td>Output high level resistance</td>
<td>ohm</td>
<td>50</td>
</tr>
<tr>
<td>DRVL</td>
<td>Output low level resistance</td>
<td>ohm</td>
<td>50</td>
</tr>
<tr>
<td>DRVZ</td>
<td>Output Z-state leakage resistance</td>
<td>ohm</td>
<td>250 Kohm</td>
</tr>
<tr>
<td>DtoA1</td>
<td>Name of level 1 DtoA interface subcircuit</td>
<td></td>
<td>DtoADefault</td>
</tr>
<tr>
<td>DtoA2</td>
<td>Name of level 2 DtoA interface subcircuit</td>
<td></td>
<td>DtoADefault</td>
</tr>
<tr>
<td>DtoA3</td>
<td>Name of level 3 DtoA interface subcircuit</td>
<td></td>
<td>DtoADefault</td>
</tr>
<tr>
<td>DtoA4</td>
<td>Name of level 4 DtoA interface subcircuit</td>
<td></td>
<td>DtoADefault</td>
</tr>
<tr>
<td>INLD</td>
<td>Input load capacitance</td>
<td>farad</td>
<td>0</td>
</tr>
<tr>
<td>INR</td>
<td>Input leakage resistance</td>
<td>ohm</td>
<td>30 Kohm</td>
</tr>
<tr>
<td>OUTLD</td>
<td>Output load capacitance</td>
<td>farad</td>
<td>0</td>
</tr>
<tr>
<td>TPWRT</td>
<td>Pulse width rejection threshold</td>
<td>sec</td>
<td>same as propagation delay</td>
</tr>
<tr>
<td>TSTOREMN</td>
<td>Minimum storage time for net to be simulated as a charge</td>
<td>sec</td>
<td>1.0 msec</td>
</tr>
<tr>
<td>TSWHL1</td>
<td>Switching time high to low for DtoA1</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSWHL2</td>
<td>Switching time high to low for DtoA2</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSWHL3</td>
<td>Switching time high to low for DtoA3</td>
<td>sec</td>
<td>0</td>
</tr>
</tbody>
</table>
INLD and OUTLD are used in the calculation of loading capacitance, which factors into the propagation delay. Refer to your PSpice user’s guide for more information.

DRVH and DRVL are used to determine the strength of the output. Refer to your PSpice user’s guide for more information.

DRVZ, INR, and TSTOREMN are used to determine which nets should be simulated as charge storage nets.

AtoD1 through AtoD4 andDtoA1 through DtoA4 are used to hold the names of interface subcircuits. Note that INLD and AtoD1 through AtoD4 do not apply to stimulus generators because they have no input nodes. Refer to your PSpice user’s guide for more information.

The switching times (TSWHLn and TSWLHn) are subtracted from a device’s propagation delay on the outputs which connect to interface nodes. This compensates for the time it takes the DtoA device to change its output voltage from its current level to that of the switching threshold. By subtracting the switching time from the propagation delay, the analog signal reaches the switching threshold at the correct time (that is, at the exact time of the digital transition). The values for these model parameters should be obtained by measuring the time it takes the analog output of the DtoA (using a nominal analog load attached) to change to the switching threshold after its digital input changes. If the switching time is larger than the propagation delay for an output, no warning is issued, and a delay of zero is used. Note that the switching time parameters are not used when the output drives a digital node.

DIGPOWER specifies the name of the power supply subcircuit the simulator calls for when an AtoD or DtoA interface is created. The default value is DIGIfPWR, which is the power supply subcircuit used by the TTL and CMOS device libraries.

For more information on how to change the default power supplies, refer to your PSpice user’s guide.

### Input/output model parameters (continued)

<table>
<thead>
<tr>
<th>Model Parameter</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSWHL4</td>
<td>Switching time high to low for DtoA4</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSWLH1</td>
<td>Switching time low to high for DtoA1</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSWLH2</td>
<td>Switching time low to high for DtoA2</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSWLH3</td>
<td>Switching time low to high for DtoA3</td>
<td>sec</td>
<td>0</td>
</tr>
<tr>
<td>TSWLH4</td>
<td>Switching time low to high for DtoA4</td>
<td>sec</td>
<td>0</td>
</tr>
</tbody>
</table>
Digital devices

Digital/analog interface devices

The simulator provides two devices for converting digital logic levels to analog voltages or vice versa. These devices are at the heart of the interface subcircuits found in dig_io.lib. These devices also provide the Digital Files interface for interfacing using external logic simulators.

Digital input (N device)

The digital input device is used to translate logic levels (typically 1s, 0s, Xs, Zs, Rs, and Fs) into representative voltage levels using series resistances. These voltages and resistances model the output stage of a logic device (like a 74LS04) and hence form a digital input to the analog circuit. The logic level information can come from two places: the digital simulator or a file. (The file can be created by hand, or can be an output file from an external logic simulator.)

The general form for a digital input device, and some of the model parameters, are different for devices driven from a file and for those driven by the digital simulation feature. The digital simulation inserts digital input devices automatically when a digital device’s output is connected to an analog component. The automatic insertion of digital input devices is discussed in your PSpice user’s guide. Examples of the devices that are inserted can be found in the dig_io.lib library file.

General form for digital simulation

N<name> <interface node> <low level node> <high level node>
+ <model name>
+ DGTLNET = <digital net name>
+ <digital I/O model name>
+ [IS = initial state]

for digital files

N<name> <interface node> <low level node> <high level node>
+ <model name>
+ [SIGNAME = <digital signal name>]
+ [IS = initial state]

Examples

N1 ANALOG DIGITAL_GND DIGITAL_PWR DIN74
+ DGTLNET=DIGITAL_NODE IO_STD
NRESET 7 15 16 FROM_TTL
N12 18 0 100 FROM_CMOS SIGNAME=VCO_GATE IS=0

Model form

.MODEL <model name> DINPUT [model parameters]
## Digital input model parameters

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHI</td>
<td>capacitance to high level node</td>
<td>farad</td>
<td>0</td>
</tr>
<tr>
<td>CLO</td>
<td>capacitance to low level node</td>
<td>farad</td>
<td>0</td>
</tr>
<tr>
<td>FILE</td>
<td>digital input file name (digital files only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORMAT</td>
<td>digital input file format (digital files only)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>S0NAME</td>
<td>state 0 character abbreviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S0TSW</td>
<td>state 0 switching time</td>
<td>sec</td>
<td></td>
</tr>
<tr>
<td>S0RLO</td>
<td>state 0 resistance to low level node</td>
<td>ohm</td>
<td></td>
</tr>
<tr>
<td>S0RHI</td>
<td>state 0 resistance to high level node</td>
<td>ohm</td>
<td></td>
</tr>
<tr>
<td>S1NAME</td>
<td>state 1 character abbreviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1TSW</td>
<td>state 1 switching time</td>
<td>sec</td>
<td></td>
</tr>
<tr>
<td>S1RLO</td>
<td>state 1 resistance to low level node</td>
<td>ohm</td>
<td></td>
</tr>
<tr>
<td>S1RHI</td>
<td>state 1 resistance to high level node</td>
<td>ohm</td>
<td></td>
</tr>
<tr>
<td>S2NAME</td>
<td>state 2 character abbreviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2TSW</td>
<td>state 2 switching time</td>
<td>sec</td>
<td></td>
</tr>
<tr>
<td>S2RLO</td>
<td>state 2 resistance to low level node</td>
<td>ohm</td>
<td></td>
</tr>
<tr>
<td>S2RHI</td>
<td>state 2 resistance to high level node</td>
<td>ohm</td>
<td></td>
</tr>
<tr>
<td>S19NAME</td>
<td>state 19 character abbreviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S19TSW</td>
<td>state 19 switching time</td>
<td>sec</td>
<td></td>
</tr>
<tr>
<td>S19RLO</td>
<td>state 19 resistance to low level node</td>
<td>ohm</td>
<td></td>
</tr>
<tr>
<td>S19RHI</td>
<td>state 19 resistance to high level node</td>
<td>ohm</td>
<td></td>
</tr>
<tr>
<td>TIMESTEP</td>
<td>digital input file step-size (digital files only)</td>
<td>sec</td>
<td>1E-91</td>
</tr>
</tbody>
</table>

* See [MODEL (model definition)](model-definition).

For more information on using the digital input device to simulate mixed analog/digital systems refer to your PSpice user’s guide.
As shown below, the digital input device is modeled as a time varying resistor from <low level node> to <interface node>, and another time varying resistor from <high level node> to <interface node>. Each of these resistors has an optional fixed value capacitor in parallel: CLO and CHI. When the state of the digital signal changes, the values of the resistors change (exponentially) from their present values to the values specified for the new state over the switching time specified by the new state. Normally the low and high level nodes would be attached to voltage sources which would correspond to the highest and lowest logic levels. (Using two resistors and two voltage levels, any voltage between the two levels can be created at any impedance.

For a digital simulation driven digital input, the parameters

\[
\text{DGTLNET} = \langle \text{digital net name} \rangle \; \langle \text{digital I/O model name} \rangle
\]

must be specified. Refer to your PSpice user’s guide for more information on digital I/O models. The digital net must not be connected to any analog devices, otherwise the automatic analog/digital interface process disconnects the digital input device from the digital net.

Digital simulation can send states named 0, 1, X, R, F, and Z to a digital input device. The simulation stops if the digital simulation sends a state which is not modeled (does not have SnNAME, SnTSW, SnRLO, and SnRHI specified) to a digital input device.

The initial state of a digital simulation driven digital input is controlled by the bias point solution of the analog/digital system. It is sometimes necessary to override this solution (for example, an oscillator which contains both analog and digital parts). The optional parameter

\[
\text{IS} = \langle \text{initial state name} \rangle
\]

can be used to do this. The digital input remains in the initial state until the digital simulation value changes from its TIME=0 value.

The model parameters FILE, FORMAT, and TIMESTEP are not used by digital simulation driven digital input devices, and only the FILE parameter is used for VIEWsim A/D driven digital inputs. For file driven digital inputs the FILE parameter defines the name of the file to be read, and the FORMAT parameter defines the format of the data in that file. The TIMESTEP parameter defines the conversion between the digital simulation’s integer timing tick numbers and the simulation’s floating-point time values:

\[
tick \text{ number} \cdot \text{TIMESTEP} = \text{seconds}
\]

\[\text{Tick number must be an integer.}\]
For a file driven or VIEWsim A/D driven digital input, the DGTLNET parameter must not be specified, but the optional parameter

SIGNAME = <digital signal name>

is used to specify the name of the digital signal in the file (or the digital net name in VIEWsim A/D). If no SIGNAME is given, then the portion of the device name after the leading N identifies the name of the digital signal.

The parameter

IS=<initial state name>

can be used as described above to override the initial (TIME=0) values from the file.

The file name DGTLPSPC is used with VIEWsim A/D to tell the simulator to get digital state values from the VIEWsim A/D interface, rather than a file.

Any number of digital input models can be specified, and both file driven and digital simulation driven digital inputs can be used in the same circuit. Different digital input models can reference the same file, or different files. If the models reference the same file, the file must be specified in the same way, or unpredictable results occur. For example, if the default drive is C:, then one model should not have FILE=C:TEST.DAT if another has FILE=TEST.DAT.

For diagnostic purposes, the state of the digital input can be viewed in Probe by specifying B(Nxxx). The value of B(Nxxx) is 0.0 if the current state is S0NAME, 1.0 if the current state is S1NAME, and so on through 19.0. B(Nxxx) cannot be specified on a .PRINT, .PLOT, or .PROBE line. (For digital simulation, the digital window of Probe provides a better way to look at the state of the digital net connected to the digital input.)
Digital output (O Device)

The digital output device is used to translate analog voltages into digital logic levels (typically 1, 0, X, R, or F). The conversion of a voltage into a logic level, models the input stage of a logic device (like a 74LS04) and hence forms a digital output from the analog circuit. The logic level information can go to two places: the digital simulation, or a file. (The file can simply be inspected manually, or can be used as a stimulus file for an external logic simulator.)

General form for digital simulation

O<name> <interface node> <reference node> <model name>  
+ DGTNET = <digital net name> <digital I/O model name>

for digital files

O<name> <interface node> <reference node> <model name>  
+ [SIGNAME = <digital signal name>]

Model form

.MMODEL <model name> DOUTPUT [model parameters]

Examples

O12 ANALOG_NODE DIGITAL_GND DO74 DGTLNET=DIGITAL_NODE IO_STD  
OVCO 17  0 TO_TTL  
05   22 100 TO_CMOS SIGNAME=VCO_OUT

Digital output model parameters

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHGONLY</td>
<td>0: write each timestep, 1: write upon change</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>CLOAD</td>
<td>output capacitor</td>
<td>farad</td>
<td>0</td>
</tr>
<tr>
<td>FILE</td>
<td>digital input file name (digital files only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORMAT</td>
<td>digital input file format (digital files only)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>RLOAD</td>
<td>output resistor</td>
<td>ohm</td>
<td>1/GMIN</td>
</tr>
<tr>
<td>S0NAME</td>
<td>state 0 character abbreviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S0VLO</td>
<td>state 0 low level voltage</td>
<td>volt</td>
<td></td>
</tr>
<tr>
<td>S0VHI</td>
<td>state 0 high level voltage</td>
<td>volt</td>
<td></td>
</tr>
<tr>
<td>S1NAME</td>
<td>state 1 character abbreviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1VLO</td>
<td>state 1 low level voltage</td>
<td>volt</td>
<td></td>
</tr>
<tr>
<td>S1VHI</td>
<td>state 1 high level voltage</td>
<td>volt</td>
<td></td>
</tr>
<tr>
<td>S2NAME</td>
<td>state 2 character abbreviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2VLO</td>
<td>state 2 low level voltage</td>
<td>volt</td>
<td></td>
</tr>
<tr>
<td>S2VHI</td>
<td>state 2 high level voltage</td>
<td>volt</td>
<td></td>
</tr>
<tr>
<td>S19NAME</td>
<td>state 19 character abbreviation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S19VLO</td>
<td>state 19 low level voltage</td>
<td>volt</td>
<td></td>
</tr>
<tr>
<td>S19VHI</td>
<td>state 19 high level voltage</td>
<td>volt</td>
<td></td>
</tr>
</tbody>
</table>
Digital devices

Digital output model parameters (continued)

<table>
<thead>
<tr>
<th>Model parameters</th>
<th>Description</th>
<th>Units</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>SXNAME</td>
<td>state applied when the interface node voltage falls outside all ranges</td>
<td></td>
<td>“?”</td>
</tr>
<tr>
<td>TIMESTEP</td>
<td>digital input file step-size</td>
<td>sec</td>
<td>1E-9</td>
</tr>
<tr>
<td>TIMESCALE</td>
<td>scale factor for timestep (digital files only)</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

* See .MODEL (model definition).

The general form for a digital output device, and some of the model parameters, are different for devices that drive a file (or VIEWsim A/D) and those that drive the digital simulation feature. The digital simulation inserts digital output devices automatically when a digital device’s input is connected to an analog component. The automatic insertion of digital output devices is discussed in your PSpice user’s guide, and examples of the devices which are inserted can be found in the dig_io.lib library file.

For more information on using the digital output device to simulate mixed analog/digital systems, refer to your PSpice user’s guide.

As shown in Figure, the digital output device is modeled as a resistor and capacitor, of the values specified in the model statement, connected between <interface node> and <reference node>. At times which are integer multiples of TIMESTEP, the state of the device node is determined and written to the specified file.

Digital output model

The process of converting the input node voltage to a logic state begins by first obtaining the difference in voltage between the <interface node> and the <reference node>. The DOUTPUT model defines a voltage range, form SxVLO to SxVHI, for each state. If the input voltage is within the range defined for the current state, no state change occurs. Otherwise, the simulator searches forward through the model, starting at the current state, to find the next state whose voltage range contains the input voltage. This state then becomes the new state. When the end of the list (S19) is reached, the simulator wraps around to S0 and continues.
If the entire model has been searched and no valid voltage range has been found, the simulator generates a simulation warning message. Further if the 0 device is interfacing at the digital simulator, and the SXNAME parameter has not been specified in the model, the simulator uses the state whose voltage range is closed to the input voltage. Otherwise it uses SXNAME as the new state.

This circular state searching mechanism allows hysteresis to be modeled directly. The following model statement models the input thresholds of a 7400 series TTL Schmitt-trigger input. Notice that the 0.8 volt overlap between the 0 state voltage range and the 1 state voltage range.

```
.model D074_STd output (  
    +s0name="0" s0vlo=1.5 s0vhi=1.7  
    +s1name="1" s1vlo=0.9 s1vhi=7.0  
    + )
```

Starting from the 0 state, a positive-going voltage must cross 1.7 volts to get out of the 0 state’s voltage range. The next state which contains that voltage is 1. Once there, a negative-going voltage must go below 0.9 volts to leave the 1 state’s range. Since no further states are defined, the simulator wraps around back to state 0, which contains the new voltage.

For a digital output driving digital simulation, the parameters

```
DGTLNET = <digital net name> <digital I/O model name>
```

must be specified. Refer to your PSpice user’s guide for more information on digital I/O models. The digital net must not be connected to any analog devices, otherwise the automatic analog/digital interface process disconnects the digital output device from the analog net.

For interfacing using digital simulation, the state names must be 0, 1, X, R, F, or Z (Z is usually not used however, since high impedance is not a voltage level). Other state names cause the simulator to stop if they occur; this includes the state ? that occurs if the voltage is outside all the ranges specified.

The model parameters TIMESCALE, FILE, CHGONLY, and FORMAT are not used for digital outputs which drive digital simulation, but the TIMESTEP is used. The TIMESTEP value controls how accurately the analog simulator tries to determine the exact time at which the node voltage crosses a threshold.

To be sure that the transition time is accurately determined, the analog simulator has to evaluate the analog circuit at intervals no larger than TIMESTEP when a transition is about to occur. The default value for TIMESTEP is 1ns, or 1/DIGFREQ (a .OPTIONS (analysis options) option) if it is larger. In many circuits, this is a much greater timing resolution than is required, and some analog simulation time can be saved by increasing the TIMESTEP value.

For digital outputs which write files, or drive VIEWsim A/D, the parameter

```
SIGNAME = <digital signal name>
```

can be used to specify the name written to the file of the digital signal (or for VIEWsim A/D, the name of the VIEWsim net). If SIGNAME is not specified, then the portion of the device name after the leading O identifies the name of the digital signal.

For digital outputs which write files, the FILE parameter defines the name of the file to be written, and the FORMAT parameter defines the format of the data written to that file.

The file name PSPCDGTL is used with VIEWsim A/D to tell the simulator to send the digital state values to the VIEWsim A/D interface, rather than a file. For VIEWsim A/D, the parameters FORMAT and CHGONLY are ignored.
The state of each device is written to the output file at times which are integer multiples of TIMESSTEP. The time that is written is the integer:

\[ \text{time} = \text{TIMESCALE} \times \text{TIME} / \text{TIMESTEP} \]

TIMESCALE defaults to 1, but if digital simulation is using a very small timestep compared to the analog simulation timestep, it can speed up the simulation to increase the value of both TIMESSTEP and TIMESCALE. This is because the simulator must take timesteps no greater than the digital TIMESSTEP size when a digital output is about to change, in order to accurately determine the exact time that the state changes. The value of TIMESSTEP should therefore be the time resolution required at the analog-digital interface. The value of TIMESCALE is then used to adjust the output time to be in the same units as digital simulation uses.

For example, if a digital simulation using a timestep of 100 ps is being run, but the circuit has a clock rate of 1us, setting TIMESSTEP to 0.1us should provide enough resolution. Setting TIMESCALE to 1000 scales the output time to be in 100 ps units.

If CHGONLY = 1, only those timesteps in which a digital output state changes are written to the file.

Any number of digital output models can be specified, and both file writing and digital simulation driving digital outputs can be used in the same circuit. Different digital output models can reference the same file, or different files. If the models reference the same file, the file must be specified in the same way, or unpredictable results occur. For example, if the default drive is C:, then one model should not have FILE=C:TEST.DAT if another has FILE=TEST.DAT.

For diagnostic purposes, the state of the digital output can be viewed in Probe by specifying B(Oxxx). The value of B(Oxxx) is 0.0 if the current state is S0NAME, 1.0 if the current state is S1NAME, and so on through 19.0. B(Oxxx) cannot be specified on a .PRINT, .PLOT, or .PROBE line. (For digital simulation, the digital window of Probe provides a better way to look at the state of the digital net connected to the digital output.)
## Digital model libraries

<table>
<thead>
<tr>
<th>File</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>7400.LIB</td>
<td>7400-series TTL</td>
</tr>
<tr>
<td>74AC.LIB</td>
<td>Advanced CMOS</td>
</tr>
<tr>
<td>74ACT.LIB</td>
<td>TTL-compatible, Advanced CMOS</td>
</tr>
<tr>
<td>75ALS.LIB</td>
<td>Advanced Low-Power Schottky TTL</td>
</tr>
<tr>
<td>74AS.LIB</td>
<td>Advanced Schottky TTL</td>
</tr>
<tr>
<td>74F.LIB</td>
<td>FAST</td>
</tr>
<tr>
<td>74H.LIB</td>
<td>High-Speed TTL</td>
</tr>
<tr>
<td>74HCT.LIB</td>
<td>TTL-compatible, High-Speed CMOS</td>
</tr>
<tr>
<td>74HC.LIB</td>
<td>High-Speed CMOS</td>
</tr>
<tr>
<td>74L.LIB</td>
<td>Low-Power TTL</td>
</tr>
<tr>
<td>74LS.LIB</td>
<td>Low-Power Schottky TTL</td>
</tr>
<tr>
<td>74S.LIB</td>
<td>Schottky TTL</td>
</tr>
<tr>
<td>CD4000.LIB</td>
<td>CD4000 devices</td>
</tr>
<tr>
<td>DIG_ECL.LIB</td>
<td>10 K and 100K ECL devices</td>
</tr>
<tr>
<td>DIG_GAL.LIB</td>
<td>GAL devices</td>
</tr>
<tr>
<td>DIG_IO.LIB</td>
<td>I/O models, AtoD and DtoA interface subcircuits, digital power supply subcircuits</td>
</tr>
<tr>
<td>DIG_MISC.LIB</td>
<td>pull-up/down resistors, delay line</td>
</tr>
<tr>
<td>DIG_PAL.LIB</td>
<td>PAL devices</td>
</tr>
<tr>
<td>DIG_PRIM.LIB</td>
<td>Digital primitives</td>
</tr>
<tr>
<td>NOM.LIB</td>
<td>master library: which references NOM_DIG.LIB,* which references each of the above libraries.</td>
</tr>
</tbody>
</table>

*Depending upon the platform being worked on, NOM.LIB references the appropriate list of libraries. For “digital only” platforms, NOM.LIB references NOM_DIG.LIB.
7400-series TTL and CMOS library files

The online Library Reference List shows, by part type and technology, each item in the library and gives the order of the pins for that function. This information is needed if a netlist is created manually. Netlists normally are generated automatically by the schematic capture package.

4000-series CMOS library

The online Library Reference List shows, by part type and technology, each item in the library and gives the order of the pins for that function. This information is needed if a netlist is created manually. Netlists normally are generated automatically by the schematic capture package.

If power supply nodes on CD4000 devices are not specified in the circuit, they can use the default power supply nodes $G_CD4000_VDD and $G_CD4000_VSS, which default to 5 volts. A new power supply can be created, and new power supply nodes can be specified to the devices in the circuit. Refer to your PSpice user’s guide for more information on specifying your own power supplies. Output drives and input thresholds are correctly modeled for power supplies between 3 and 18 volts. Currently, propagation delays do not vary using supply voltages. For correct propagation delays at supply voltages other than 5 volts, the timing models in cd4000.lib have to be modified.
Programmable array logic devices

Using a PLD from the library is just like using any other logic device from the library, except that the simulator has to be told the name of the JEDEC file which contains the program for the part. A TEXT parameter name JEDEC_FILE is used to specify the file name, as shown in the following example:

```
X1 IN1 IN2 IN3 IN4 IN5 IN6 IN7 IN8 IN9 IN10 IN11 IN12
+ IN13 IN14
+ OUT1 OUT2 OUT3 OUT4
+ PAL14H4
+ TEXT: JEDEC_FILE = “myprog.jed”
```

This example creates a 14H4 PAL which is programmed by the JEDEC file myprog.jed.
# Customizing device equations

<table>
<thead>
<tr>
<th>Introduction to device equations</th>
<th>Specifying new internal device structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Making device model changes</td>
<td>Recompiling and linking the device equations option</td>
</tr>
<tr>
<td>Changing the device equations</td>
<td>Simulating with the device equations option</td>
</tr>
<tr>
<td>Adding a new device</td>
<td></td>
</tr>
</tbody>
</table>
Introduction to Device Equations

The purpose of the Device Equations option is to change the built-in model equations for one or more of the semiconductor devices (GaAsFET, Diode, Junction FET, MOSFET, and Bipolar transistor). This means you can extend PSpice to support user-defined or proprietary native device models.

This option is not an addition to PSpice: it is a different packaging of the program that includes the source code for the device model subroutines. You need a Device Equations license to modify and extend PSpice code, but you do not need a Device Equations license to use the modified code.

There are several kinds of changes that can be made using the Device Equations option. These include, in ascending order of complexity:

- Changing a parameter name
- Giving a parameter an alias
- Adding a parameter
- Changing the device equations
- Adding a new device
- Specifying new internal device structure

You need a supported C++ compiler to compile Device Equations extensions; for Windows 95/98 and NT, you need Microsoft Visual C++ 32-bit Compiler 4.2 or later.

Device Equations extensions are implemented using a dynamic-link library, which means you can share your models with other users by distributing just a DLL.

If you want to run PSpice on Windows 95 or NT with a Device Equations DLL developed by someone else, then you do not need a compiler or a Device Equations license. Just copy the DLL into the directory with your PSpice program file. For more information, see Simulating with the Device Equations option.
Making device model changes

To get started, look at the files M.H and MOS.C, which implement the MOSFET equations. The other devices have similar structures.

M.H contains two important data structure definitions:

- the structure for the MOS transistor \( \text{struct m}_\)  
- the structure for the MOS model \( \text{struct M}_\)

During read-in, the simulator creates a copy of the transistor structure for every MOSFET in the circuit and a copy of the model structure for every .MODEL statement of type NMOS or PMOS. The transistor structure is set up using information particular to that transistor, such as the nodes to which it is connected, its length and width, and the locations of its entries in the circuit’s conductance matrix. All parameters of the model structure are set up using the values from the .MODEL statement, if one exists; otherwise, the default values are used.

The transistor structure corresponds to the LOC, LOCV, and LX tables in U.C. Berkeley SPICE2. The model structure corresponds to the LOC and LOCM tables in SPICE.

\[ \text{Do not change the transistor structure (struct_m), except when changing the internal device topology. It is included only to allow compiling of MOS.C.} \]

The simulator needs to associate each entry in the model structure with a model parameter name (and default value) in the .MODEL statement. You can accomplish this by using the \text{ASSOCIATE} macro. Just below the model structure in M.H there is a list of all the parameters, each in an \text{ASSOCIATE} macro. The occurrence of \text{ASSOCIATE} binds together the structure entry, the parameter name, and the default value. The read-in section of the simulator uses this information to parse the .MODEL statement.
Changing a parameter name

This is the easiest change. Find the parameter in the list of ASSOCIATE macros. Change the parameter’s name (last item on the line) and/or the default value (middle item). The names and defaults of the model parameters that are supplied can be changed, as well as those parameters that are added.

When the simulator runs, it prints the parameter values for each .MODEL statement unless the NOMOD option is used in the .OPTIONS statement. Normally only parameters which have not been defaulted are listed. A parameter can be forced to be listed, whether or not it has been defaulted, by preceding its name using an asterisk (*). For example, VTO is listed that way in M.H.

Giving a parameter an alias

Sometimes a parameter requires an alternate name (an alias). Several bipolar model parameters, such as ISE, already have alternate names. The alias for ISE is C2. Look in Q.H at the occurrences of the parameters ISE and C2 in the ASSOCIATE macros for an example of how this is accomplished. There is only one entry in the model structure (Q_ise) for the parameter, but there are two ASSOCIATE entries. This means that either name (ISE or C2) on the .MODEL statement can put a number into the structure entry Q_ise.

When model parameters are listed, the first name found in the ASSOCIATE list (searching downward) is the name which is echoed on the output.

Insert the new name first if it is the name to be printed.

Adding a parameter

Adding a parameter is probably the most common case. The parameter must be added to both the model structure (e.g., struct M_) and the corresponding ASSOCIATE list. It is recommended to follow the OrCAD naming convention (e.g., M_wd and M_vto), but it is not required.

Model parameters are set forth as pairs of elements instead of simple floating point values. This is to provide the use of expressions for model parameters. Because of this, when adding a parameter (for example, M_new), the following line is required:

```
MXPR( M_new, Mx_new );
```

instead of

```
float M_new;
```

Do not modify the value of the Mx_new structure element.

The read-in mechanism can handle expressions for user-added parameters. By the time the model code is called, the expressions have been evaluated and their value placed in the appropriate fields. See the include file m.h for further examples and comments.
Customizing device equations

When the simulator is doing a read-in, model parameters are listed for each .MODEL statement (unless NOMOD has been specified on the .OPTIONS statement). Normally, only those parameters that have not been defaulted are listed. A parameter can be forced to be listed, even if it has been defaulted, by preceding its name using an asterisk (*) in the ASSOCIATE macro. For instance, VTO in M.H is listed in that manner.

The default value, OMITTED, is used by the simulator to force the calculation of a parameter’s value during read-in. For instance, VTO is calculated from other values if it is not given a value. These calculations are built into the read-in and are fixed. OrCAD recommends that parameters that you add be given a normal default value and not be computed by using OMITTED.

Once the parameter has been added, the model structure becomes one parameter longer, and the read-in section of PSpice places a value in its entry. The parameter can now be used in the device code (e.g., MOS.C).

Changing the device equations

The device equations are in the file that has the same name as the type of device (DIODE.C, BJT.C, JFET.C, MOS.C, GASFET.C). The code in these subroutines use the model parameters and the device’s terminal voltages to calculate the branch currents and conductances, and, during transient analysis, the terminal charges and branch capacitances. These equations are neither simple nor easy. A good understanding of U.C. Berkeley’s SPICE2G is recommended before making such a change. Two useful references are:


which are available from:

Software Distribution Office
EECS/ERL Industrial Liaison Program
205 Cory Hall #1770
University of California
Berkeley, CA 94720-1770
(510) 643-6687
Customizing device equations

Making device model changes

Functional subsections of the device source file

The code in each of the device source files is arranged into separate functional subsections. Each subsection occurs at least once, but can occur several times for devices that have more than one level. The subsections required are outlined below.

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization</td>
<td>This consists of locating and binding the device instance and its model, initializing any local variables, and obtaining appropriate values for the device branch voltages. The branch voltages (e.g., vds, vgs) are set differently depending upon whether there are user-specified initial conditions (using IC= or .IC), and on whether the present Newton Raphson cycle has finished or not.</td>
</tr>
<tr>
<td>Computing new nonlinear branch voltage:</td>
<td>This is needed to monitor progress towards a Newton Raphson solution.</td>
</tr>
<tr>
<td>Test if the solution has changed:</td>
<td>If there is not significant change bypass the rest of the computation. Otherwise, continue.</td>
</tr>
<tr>
<td>Limit any nonlinear branch voltages:</td>
<td>This code uses the macro PNJLIM() to insure that the branch voltages are in the appropriate operating region.</td>
</tr>
<tr>
<td>Compute currents and conductances:</td>
<td>This is the meat of the Device Equations code, and involves obtaining all the branch currents (e.g., ibs, ibd) as well as all the derivatives to be used in the conductance matrix.</td>
</tr>
<tr>
<td>Charge calculations:</td>
<td>Internal charges are calculated and updated.</td>
</tr>
<tr>
<td>Check convergence:</td>
<td>Check to see if the nonlinear device branches now have values that are within a small tolerance range of those obtained in the last repeat cycle, and set a return flag to signal whether the device converged.</td>
</tr>
<tr>
<td>Load the current vector and conductance matrix:</td>
<td>The macro Y_MATRIX () is used to obtain handles to the proper matrix elements, and the elements are assigned their values based on the present evaluation of the device equations and derivatives.</td>
</tr>
</tbody>
</table>

SPICE2G is written in FORTRAN, whereas PSpice is in C. For the device subroutines, as much correspondence as possible has been maintained between the two. Because of FORTRAN, SPICE kept integer and real numbers in different tables: NODPLC (indexed by LOC) and VALUE (indexed by LOCV or LOC). In PSpice, these have been combined into one structure (e.g., struct m_).

The state vector information is constructed somewhat differently, though the overall pattern is similar. In SPICE the state vector information is kept in a set of vectors in VALUE. There is one vector for each time point “remembered” (from 4 to 7, depending on the order of the integration method). Each device’s LOC table contains an offset, LX, to its portion of the information in each state vector. In PSpice the number of state vectors is fixed, and each device’s state information is kept in its own device structure (e.g., struct m_).

For example, for MOSFETs the state vectors are an array, struct msv_def m_sv[MSTVCT] in struct m_, MSTVCT is the number of state vectors and is defined in TRAN.H to be equal to 4. The definition of msv_def (also in M.H) lists the various currents, conductances, charges, and capacitances that are in the state vector. Finally, M.H contains a set of #defines, which allows accessing of the entries to the state vectors by name. It is these (uppercase) names
Customizing device equations

Making device model changes

which are then used in MOS.C. This may seem like a roundabout way of constructing the state vector information, but the actual usage (in MOS.C) is quite straightforward and is similar to that in SPICE.

Adding a new device

The Device Equations option does not allow the addition of an entirely new device. However, in many cases the same thing can be achieved by making use of an existing device.

Suppose, for example, that a lightning arrester device is to be added. The lightning arrester has two terminals, therefore it can be built into the diode equations, because the diode also has two terminals. This means that in the circuit (.CIR) file the lightning arresters would use the letter D to start and would refer to a .MODEL statement of the type D.

At first glance it appears that this would preclude using diodes in circuits, since they have been replaced by lightning arresters. This problem is avoided by keeping all the diode model parameters, adding the lightning arrester parameters, adding a LEVEL parameter, and giving the LEVEL parameter a default of 1. In the diode subroutine (in DIODE.C), a large if test would select all the old diode code if LEVEL=1 and all the new lightning arrester code otherwise. The new LEVEL parameter would switch between diode and lightning arrester.

This approach can be extended to as many devices as wanted. This could be:

- LEVEL=1 as a diode
- LEVEL=2 as a lightning arrester
- LEVEL=3 as a gas discharge tube

And so on. The restriction is that all of the devices added to the diode must have two terminals. If the device to be added has three terminals, it must be built into a three terminal device, such as the JFET. The highest number of terminals that can be modeled is four, using the MOSFET. There is not a good way to add devices, such as pentodes, that have five or more terminals.
Specifying new internal device structure

You may want to change the topology of a device in order to accommodate a more elaborate set of parasitic resistances and/or capacitances. To do this requires that positions in the conductance matrix be assigned to include the terms that the additional equations generate. This requires five steps:

1. Ensuring that all of the new internal nodes and matrix conductance terms are added to the device structure in the device header file
2. Allocating the new matrix elements
3. Providing handles to access the new matrix elements and to bind the nodes to the branches
4. Including logic, if needed, to support device model parameter checking and updating
5. Adding the new device equations to the device code

Example

This process can be illustrated by looking at the PSpice JFET and GaAsFET devices, as shown below. The topologies of these two devices are nearly identical, except that the GaAsFET has an additional internal capacitance, CDS, between the source and drain, and an additional internal resistance, RG, at the gate. This gives the GaAsFET topology one additional node where RG joins the rest of the structure and two additional internal branches.
Customizing device equations

Procedure

Step one: editing the device header file

These differences are reflected in the device structure definitions in J.H and B.H. Each of the device nodes is given a name and declared to be of type CKT_IDX.

The JFET device structure, _j_, lists the two internal nodes _j_d and _j_s, while the GaAsFET device structure, _b_, has three internal nodes _b_d, _b_s, and a new one, _b_g. The two additional branches in the GaAsFET require three new matrix conductance terms.

The conductance terms are declared type MTX_IDX and are listed immediately following the internal nodes.

The JFET has a term _j_GG, which appears on the matrix diagonal for the external gate node.

The GaAsFET has an additional gate node which requires one additional matrix diagonal conductance term, _b_gg, along with two off-diagonal conductance terms, _b_Gg and _b_gG. These are used by the source code in GASFET.C to designate where the conductance terms associated with RG go when the matrix is loaded. CDS doesn’t need any additional nodes or matrix terms because the items required are already in place to accommodate the parallel current source, id.

With the nodes and conductance terms taken care of in the device header file, the first step is completed.

Step two: setting up memory allocation for the new matrix elements

You can set up memory allocation to properly incorporate the new equations into the conductance matrix by modifying DEMATPTR.C. In this file are the functions _j_MatPtr() and _b_MatPtr(). These functions call the function Reserve() once for each conductance matrix term that was declared in the header file. For instance, when _b_gg, _b_Gg, and _b_gG are added for the GaAsFET, these require corresponding code in _b_MatPtr() as follows:

flag &= Reserve (ng,ng);
flag &= Reserve (nG,ng);
flag &= Reserve (ng,nG);

The arguments _ng and _nG are local variables that serve as aliases for the respective device nodes, _b_g and _b_G, and are assigned at the beginning of _b_MatPtr() as follows:

ng = bloc -> b_g;
nG = bloc -> b_G;

Step three: binding the nodes and branches

The mechanics of step three, binding the nodes and branches, are very similar to the mechanics of step two. This time DEMATLOC.C is modified. The functions of interest are _j_MatLoc() and _b_MatLoc(), and they now call _Indxcl() instead of Reserve(). The GaAsFET again has three more lines of code:

flag &= _Indxcl (&(bloc->b_gg),ng,ng);
flag &= _Indxcl (&(bloc->b_Gg),nG,ng);
flag &= _Indxcl (&(bloc->b_gG),ng,nG);
Step four: handling model parameters

Step four, handling model parameters, is basically the same as it would be for a case not involving topology changes, with one significant exception: this requires handling the case where the parasitics associated with an internal node can be zero. In this case the node must be generated conditionally. An instance of this is the GaAsFET internal resistance $R_G$. If $R_G$ is zero, the parasitic resistance between the internal node $b_g$ and the external node $b_G$ can be removed from the circuit. This is accomplished in the function $b_{\text{AddInternalNodes}}()$ in DEMODCHK.C, using the following line of code:

```c
INTERNAL_NODE(P->B_rg,b_g,b_G);
```

`INTERNAL_NODE()` is a macro that performs the required logic, depending on whether the model parameter $B_{rg}$ is zero or not. The other two calls to this macro in $b_{\text{AddInternalNodes}}()$ correspond to the RD and RS resistances that also exist for the JFET.

Step five: implementing the new device equations

The final step does not involve any further topological considerations and is carried out just as it would be if the device internal topology weren’t being changed.
Recompiling and linking the Device Equations option

The object and source files needed to create the Device Equations DLL are installed in a directory called DEVEQ. The MSVC++ 4.2 project files, deveq.mdp and deveq.mak, are included to compile and link the DLL.

For information on obtaining the Microsoft compiler, contact Microsoft Corporation directly.

To create a new deveq.dll

1 Load deveq.mdp into the Visual C++ development environment.
2 From the Build menu, select Build Deveq.dll.
   The project supports debug and release versions of the build target.
3 After deveq.dll is built, copy it to the directory that contains pspice.exe.

Personalizing your DLL

The function DLLMain() in deveqdll.c contains the following line of code:

DEVEQVERSIONINFO("",VERSIONNUM);

To personalize your DLL, change the two arguments, as in:

DEVEQVERSIONINFO("(c)Copyright 1998\nMyCorp","7.2.1");
Simulating with the Device Equations option

After you obtain a working Device Equations DLL, place it in the directory that contains pspice.exe.

If your PSpice license has the Device Equations option, PSpice will locate and load deveq.dll when you start the program. The code in the DLL will be substituted for the device model code that ships with the plain version of PSpice. The title bar will indicate that PSpice is using the DLL by showing the program name as PSpice/DE. The presence of the DLL is also noted in the About box and in the .out file.

If PSpice doesn’t find the DLL, it runs as the normally configured PSpice.

New! Selecting which models to use from a Device Equations DLL

You can tell PSpice which device models to use from a custom DLL by adding an entry to the pspice.ini configuration file; for any device type you do not specify, PSpice uses the normally configured PSpice models.

To specify which models to use from a custom DLL

1. In a standard text editor (such as Notepad), open pspice.ini, located in the directory with your PSpice program file.

2. Find the [ORCAD] section and add this line to the section:
   USE_DEVEQ_MODELS="<device letters>"

   where <device letters> is any or all of the following:

<table>
<thead>
<tr>
<th>For this device type...</th>
<th>Use this device letter...</th>
</tr>
</thead>
<tbody>
<tr>
<td>GaAsFET</td>
<td>B</td>
</tr>
<tr>
<td>Diode</td>
<td>D</td>
</tr>
<tr>
<td>Junction FET</td>
<td>J</td>
</tr>
<tr>
<td>MOSFET</td>
<td>M</td>
</tr>
<tr>
<td>Bipolar transistor</td>
<td>Q</td>
</tr>
</tbody>
</table>

   For example, to use all of the possible device models from your custom DLL, type the following:
   USE_DEVEQ_MODELS="BDJMQ"

3. Save pspice.ini.

4. Start PSpice and run a simulation.
Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABM</td>
<td>analog behavioral modeling</td>
</tr>
<tr>
<td>AKO</td>
<td>“A Kind Of” symbol. Symbols must either contain graphics or refer to an AKO symbol. The AKO defines the symbol in terms of the graphics and pins of another part. Both must exist in the same Symbol Library file.</td>
</tr>
<tr>
<td>alias</td>
<td>An alias relates local schematic names for parts and signals to netlist names (simulation devices and nodes). An alias is an exact electrical equivalent that can be used to reference a symbol. A command that sets up equivalences between pin names or net names and node names. As a command, it is the setup equivalences between node names and pin names or net names.</td>
</tr>
<tr>
<td>annotation</td>
<td>Annotation is a means by which parts are labeled when they are placed, either automatically or manually.</td>
</tr>
<tr>
<td>annotation symbol</td>
<td>An annotation symbol has no electrical significance, and is used to clarify, point out, or define items on the schematic.</td>
</tr>
<tr>
<td>argument</td>
<td>A value or an expression used with an operator or passed to a subprogram (subroutine, procedure, or function).</td>
</tr>
<tr>
<td>attributes</td>
<td>Attributes are special characteristics (a name and an associated value) contained in a part instance or definition. For example, a MOSFET may contain specific length and width parameters which are represented as attributes on the symbol or part. Attributes may be changed through the Schematic Editor and/or the Symbol Editor.</td>
</tr>
<tr>
<td>block</td>
<td>A block is a user defined rectangle placed on a schematic. It is used to represent or hold the place for a collection of circuitry. The block is treated as a black box by Schematics. Schematics is aware of the connections going into and out of the block, but ignores the contents of the block until the netlist is generated.</td>
</tr>
<tr>
<td>bus</td>
<td>A bus is a collection of homogeneously named signals.</td>
</tr>
<tr>
<td>call</td>
<td>To transfer a program execution to some section of code (usually a subroutine of some sort), while saving the necessary information to allow execution to resume at the calling point when the call section has completed execution.</td>
</tr>
<tr>
<td>circuit</td>
<td>A circuit is a configuration of electrically connected components or devices.</td>
</tr>
<tr>
<td>comment</td>
<td>A statement written into a program for documentation purposes only and not for any functionality purposes.</td>
</tr>
<tr>
<td>compiler</td>
<td>Translates between high-level computer language understood by humans and machine language that is understood by computers.</td>
</tr>
<tr>
<td>component</td>
<td>A device or part employed in a circuit to obtain some desired action. See package.</td>
</tr>
<tr>
<td><strong>connector</strong></td>
<td>A connector is a physical device that is used for external connections to a circuit board.</td>
</tr>
<tr>
<td><strong>construct</strong></td>
<td>A computer program statement that produces a predetermined effect.</td>
</tr>
<tr>
<td><strong>current source</strong></td>
<td>A current source can be an ideal current source (no limit on the supply voltage) or a voltage source with a series resistor.</td>
</tr>
<tr>
<td><strong>defined function</strong></td>
<td>A computer instruction specifying the operation to be done with predetermined limits.</td>
</tr>
<tr>
<td><strong>declarative statement</strong></td>
<td>A computer source program instruction specifying the size, format, and kind of data elements and variables in a program for a compiler.</td>
</tr>
<tr>
<td><strong>device</strong></td>
<td>A simple or complex discrete electronic component. Sometimes, a subsystem employed as a unit and, therefore, thought of as a single component. See package.</td>
</tr>
<tr>
<td><strong>DIBL</strong></td>
<td>drain-induced barrier lowering (MOSFET device)</td>
</tr>
<tr>
<td><strong>dot command</strong></td>
<td>A type of formatting command typed into a document that is preceded by a period (dot) to distinguish from other syntax text.</td>
</tr>
<tr>
<td><strong>doping tail</strong></td>
<td>A changing amount of impurity in a semiconductor device. It is observed as a change in the bulk resistance of the semiconductor material.</td>
</tr>
<tr>
<td><strong>ELSE</strong></td>
<td>An operation used in BASIC computer programming. It specifies the operation to be performed if the conditions given in the same program line didn’t occur.</td>
</tr>
<tr>
<td><strong>flicker noise</strong></td>
<td>A repeating low-frequency noise.</td>
</tr>
<tr>
<td><strong>Fourier analysis</strong></td>
<td>A mathematical method of transforming a function in such a way that the data of the function is retained but the representation of that data is changed. It is used to simplify the reduction of the data.</td>
</tr>
<tr>
<td><strong>FSTIM</strong></td>
<td>digital file stimulus device</td>
</tr>
<tr>
<td><strong>gate</strong></td>
<td>A gate is a subset of a package, and corresponds to a part instance. An electronic switch that follows a rule of Boolean logic.</td>
</tr>
<tr>
<td><strong>glitch</strong></td>
<td>An unwanted transient that recurs irregularly in the system.</td>
</tr>
<tr>
<td><strong>global temperature</strong></td>
<td>Universally applied temperature (to all elements of a circuit)</td>
</tr>
<tr>
<td><strong>global parameter</strong></td>
<td>Universally applied parameter (to all elements of a circuit)</td>
</tr>
<tr>
<td><strong>icon</strong></td>
<td>A small graphics image displayed on the screen to represent an object that can be manipulated by the user.</td>
</tr>
<tr>
<td><strong>IF</strong></td>
<td>An operation used in BASIC computer programming. It specifies an IF-THEN operation to be performed when a condition has changed from what was expected in a program line.</td>
</tr>
<tr>
<td><strong>ISAS</strong></td>
<td>independent current source and stimulus</td>
</tr>
<tr>
<td><strong>included file</strong></td>
<td>A smaller file that is read into a larger source-code file at a specific spot and becomes part of a statement within the larger source-code file.</td>
</tr>
<tr>
<td><strong>instance name</strong></td>
<td>A name of an object in an object oriented programming. It is a unique name for a part instance.</td>
</tr>
<tr>
<td><strong>instantiate</strong></td>
<td>To create an instance of a class in object oriented programming.</td>
</tr>
</tbody>
</table>
invocation  To start a software program by invoking an initial power from a higher power

invoke  To call or activate; used in reference to commands and subroutines.

ionization knee  A bend in the response curve where ionization starts.

IS temperature  The temperature of the JFET and other transistor types junction saturation
current or the input leakage current

iteration  A repeating series of arithmetic operations to arrive at a solution.

Jiles-Atherton model  A state equation model rather than an explicit function for an inductor

junction  A junction graphically indicates that wires, buses, and/or pins are electrically
connected.

keyword  The significant word in a syntax statement that directs the process of the
operation.

labels  Is a word or symbol used to identify a file or other element defined in a
computer program.

LIBPATH  A variable that specifies the directory that the model library is in, and is first
set in the msim.ini file.

link  A branch instruction, or an address in such an instruction, used to leave a
subroutine to return to some point in the main program.

lot tolerance  The tolerance of a group of items taken as one unit.

lsb  least significant bit

metafile  A file that contains or defines other files.

mobility  movement of electrons in semiconductor devices such as MOSFETs

model library  consists of analog models of off-the-shelf parts that can be used directly in
circuits that are being developed

mouse  A common pointing device used in a windows environment. The physical
movement of the mouse will move the pointer (cursor) on the screen.

msb  most significant bit

msim.ini  The MicroSim configuration file that has the default elements that are used to
complete a simulation.

nesting  The embedding of one construct (such as a table in a database; a data
structure, a control structure) inside another—for example, a nested procedure
is a procedure declared within a procedure.

NETLIST  The netlist provides the circuit definition and connectivity information in
simulation netlist format.

NODESET  A nodeset symbol contains one or two pins, permitting you to initialize a node
voltage for simulation.

NOREUSE flag  A piece of information that tells the simulator that the automatic saving and
restoring of bias point information between different temperatures, Monte
Carlo runs, worst-case runs, or parametric analyses is suppressed. It is one of
the options in the .OPTIONS (analysis options) command.
Glossary

NOSUBCKT  A variable that tells the simulator not to save the node voltages and inductor currents for subcircuits.

NUMDGT  An option that tells the simulator the number of digits that will be printed for the analog values. It is one of the options in the `.OPTIONS (analysis options)` command.

object  A variable comprising both routines and data that is treated as a discrete entity, in object-oriented programming.

operator  A symbol (mathematical, as an example) or other character indicating an operation that acts on one or more elements.

OUTPUT ALL  An option that asks for an output from the sensitivity runs, after the nominal (first) run. The output from any run is governed by the `.PRINT (print), .PLOT (plot), and .PROBE (Probe)` command in the file. If OUTPUT ALL is omitted, then only the nominal and worst-case runs produce output. OUTPUT ALL ensures that all sensitivity information is saved for Probe.

package  A package is an enclosure for an electronic device or subsystem. A physical device consisting of one or more gates.

page  A page may contain both parts (represented by symbols), port instances, connectors, and annotation symbols. A page may or may not have a title. Each schematic page represents a single page of a circuit design.

parameter  A value that is given to a variable for programing.

part  A part is an electrical component that is represented by a schematic symbol. The term refers to the logical, rather than the physical, component.

part definition  See symbol.

part instance  A part instance refers to an occurrence of a symbol in a schematic.

pin  Pins are contained in parts, ports, and offpage connectors. Parts can contain multiple pins. Each part contains specific pin names associated with the part. Pins may connect to a wire, a bus, or another pin.

pin current  The current that flows into or out-of a defined pin.

POLY  Specifies the number of dimensions of the polynomial.

port  A port provides connectivity across schematic pages. A port provides the anchor for a single pin. Ports are chosen from library files, placed, moved, and deleted in the same way as are parts. Ports may have multiple connections. Ports consist of three types: global, interface, and offpage.

run  The execution of a computer routine or operation.

SCBE  substrate current induced body effect (MOSFET device)

schematic  A schematic consists of the following components: one or more pages, a set of symbols representing local part definitions or parts in a library file, and/or text.

setpoint  A setpoint provides a graphical way of introducing `.IC (initial bias point condition)` or `.NODESET (set approximate node voltage for bias point)` commands for each instance of a symbol. These commands set one or more node voltages for the bias point calculation.
SIMLIBPATH  A variable that defines the environment that the simulator is working in (path to the directory that the library is in).

simulation  The use of a mathematical model to represent a physical device or process.

skipbp  (skip bias point)

statement  The smallest executable entity within a programming language. In general, each line of a program is an individual statement and is considered an individual instruction. (Examples: command statements, option statements, control statements, assignment statements, comment statements.)

Statz model  A GaAsFET model

subcircuit  A small collection of components working together to perform a task.

symbol  A symbol consists of the graphical representation of a logical or physical electronic part on the schematic page, and its definition. Symbols can be created either for a specific schematic or extracted from a library file, and may contain schematic pages nested within them.

syntax  The grammar of a particular computer language, with rules that govern the structure and content of the statement.

TEXTINT  A function which returns a text string which is the integer value closest to the value of the <value or expression>; (<value or expression> is a floating-point value)

tick number  The number generated from a regular recurring signal emitted by a clocking circuit, or from the interrupt generated by this signal.

TOM model  a GaAsFET device

VARY BOTH  The default option is VARY BOTH. When VARY BOTH is used, sensitivity to parameters using both DEV and LOT specifications is checked only with respect to LOT variations. The parameter is then maximized or minimized using both DEV and LOT tolerances for the worst-case. All devices referencing the model have the same parameter values for the worst-case simulation.

VARY DEV  See VARY BOTH

VARY LOT  See VARY BOTH

VTO temperature  The temperature of the JFET or MOSFET device when there is zero-bias threshold (pinchoff) voltage.

window  An area on the screen in a graphical computer interface that contains instructional documentation or a message.
* (comment), 99
; (in-line comment), 100

**Numerics**
4000-series CMOS library, 333
7400-series TTL and CMOS libraries, 333
74181 model, 294
74393 subcircuit example, 249

A
A/D and D/A converters, 286
ABM, 137
  defined, 347
  polynomial transfer function, 138
absolute value (ABS), xix
ABSTOL (.OPTIONS), 61
.AC, 32
AC analysis, 32
ACCT (.OPTIONS), 59
ACOS(x), xix
ADC, 287
AFFECTS, 302
air-gap, 165, 166
AKO, 112, 155, 173, 185, 207, 217
  defined, 347
alias, 338
  defined, 347
.ALTER, 103
amplitude, peak, 150, 151
analog devices, 53
  passive
  semiconductor
analog parts
  breakout parts, 129, 163, 172, 216
  capacitors, 216
  ideal switches, 220, 233
  inductor coupling (KBREAK), 163
  inductors, 216
  resistors, 216
  semiconductor parts, 132, 177
analog-to-digital converter, 51
analyses
  AC, 32
  bias point, 58
  DC, 34
  Fourier, 41
  Monte Carlo, 47
  noise, 56
  parametric, 79
  sensitivity, 78
  sensitivity/worst-case, 95
  temperature, 87
  transient, 90
analysis options
  flag options, 59
AND, 256
AND3, 259
anhydrous, 165
annotation
  defined, 347
annotation symbol
  defined, 347
AO, 256
arc tangent (ATAN and ARCTAN), xix
arccosine function, xix
ARCOS(x), xix
arcsine, xix
arctangent, xix
argument, 85, 86, 236, 347
arithmetic expressions, xx
ASIN(x), xix
ATAN2, xix
attributes
  definition, 53, 60, 347

B
base-
emitter voltage, 73
terminal abbreviation, 71
behavioral primitives, 249, 291
cost effective check, 304
logic expression, 292
pin-to-pin delay, 295
bias point, 58
.NODESET, 55
SKIPBP, 90
small-signal, 43
biaspoint
transient, 43
bidirectional delay line, 223
bidirectional transfer gates, 248, 261
binary notation format, 313
bipolar transistor, 204
summary, 105, 107
bipolar transistor model
quasi-saturation effect, 212
bipolar transistor, PNP, 51
block, 236, 237
BOOLEAN, 296, 297, 304
Boolean expression IF, xix
breakout parts, 129, 172, 216
inductor coupling, 163
BSIM3 model
advanced parameters, 181
BUF, 256
BUF3, 259
bulk, MOSFET terminal (substrate), 71
bus, 63
defined, 347

C
call, 69, 85, 106, 236, 347
CAP device model, 51
capacitor, 51, 128
summary, 105, 107
capacitors, 216
t voltage coefficients, 102
CASE, 300
CHANGED
reference function, 298
CHANGED_HL
reference function, 298, 300
CHANGED_LH
reference function, 298
CHEBYSHEV, 136
CHGTOL (.OPTIONS), 61
circuit, 106, 128, 236, 237, 347
circuit topology, 77
CLEAR, 305
CLOCK, 304
CLRBAR, 300
Cohen, Ellis, 339
collector, 71
cmd
reference, 30
syntax format, xvii
command files
Probe, xxii
command line options
PLLogic, xxvii
PSpice, xxvii
PSpice A/D, xxvii
comment, 99, 347
bias point files, 46
included files, 44
in-line, 100
comment line, 99
common simulation data file (CSDF), 69
compiler, 336, 345, 347
compiling, 345
complex digital devices, 291
component, 106, 170, 215, 237, 347
conductance, 61
conductance matrix, 337
connectors, 348
CONSTRAINT, 294, 302, 304, 308
PSpice messages, 63
constraint check, 304
FREQ, 307
GENERAL, 307
SETUP_HOLD, 304
WIDTH, 306
continuation line, 277, 281, 283
conventions, 30
expression, 30
numeric value, 30
convergence hazard, 63
convergence problems, 77
corrections, 103
converters, 286, 289
convolution, 225
CORE device model, 51
core model, 167, 170
cosine (COS), xix
CPTIME (.OPTIONS), 61
CPU time, 61
current source, 32, 348
   EXP parameters, 144
   PULSE parameters, 145
   SIN parameters, 151
   SSFM parameters, 150
current-controlled
   current source, 105, 107, 141
   switch, 51, 105, 107, 232
   voltage source, 105, 107, 141
topology, 342
device equations
   adding a new device, 341, 342
   adding a parameter, 338
   changing a parameter’s name, 338
   changing the device equations, 339
   example, 342
   giving a parameter an alias, 338
   making device model changes, 337
   recompiling and linking, 345
device temperatures, customized, 53
devices, 105, 107
   A/D converter, 286
   bipolar transistor, 204
   capacitor, 128
   complex digital, 291
   current-controlled current source, 141
   current-controlled switch, 232
   current-controlled voltage source, 141
   D/A converter, 286
   digital input, 324
   digital output, 328
   digital primitive, 105
   digital stimulus, 105
   digital-to-analog interface, 324
   diode, 131
   GaAsFET, 110
   independent current source & stimulus, 142
   independent current source & stimulus
      (sinusoidal waveform), 151
   independent voltage source & stimulus, 142
   inductor, 170
   inductor coupling, 105
   inductor coupling (transformer core), 160
   input/output model, 322
   insulated gate bipolar transistor, 237
   interface, 246
   JFET, 153
   MOSFET, 105, 108, 174
   passive, 53
   primitives, 247
   programmable array logic, 334
   resistor, 215
   semiconductor, 53
   stimulus, 246
   subcircuit instantiation, 236
   subcircuits, 105

D
D device model, 51
DAC, 289
damping factor
   current source, 151
data
   file, xxvi
   .DC, 34
   DC analysis, 34
   DC gain, 89
   DC sweep, 34
   Ddt(x), xix
declarative statement, 348
   DEFAD (.OPTIONS), 61
   DEFAS, 61
default temperature (TNOM), 62
   defined function, 37, 42, 348
   DEFL (.OPTIONS), 61
   DEFW (.OPTIONS), 61
   DEG, 137
   DELAY, 136
delay, 299
   line, 248, 274
   values, 251
   DEV tolerance, 52, 98
   DEVEQU, 345
device, 178, 179, 181, 348
   header file, 342
   model change, 337

355
transmission line, 223
transmission line coupling, 105, 109, 160
voltage-controlled current source, 136
voltage-controlled switch, 219
voltage-controlled voltage source, 136

DFF, 265
DIBL, 181, 189, 190, 348
differential function, xix
DIGDRVF (.OPTIONS), 61
DIGDRVZ (.OPTIONS), 61
DIGERRDEFAULT (.OPTIONS), 61, 308
DIGERRLIMIT (.OPTIONS), 61, 308
DIGFREQ (.OPTIONS), 61
DIGINITSTATE (.OPTIONS), 61
digital delay line, 51
digital input, 51, 324
digital input model parameters, 325
  C (capacitance), 325
  FILE, 325
  FORMAT, 325
  Sn (state "n"), 325
  TIMESTEP, 325
digital libraries, 332
digital output, 51, 328
  .VECTOR, 92
digital output model parameters, 328
  CHGONLY, 328
  CLOAD, 328
  FILE, 328
  FORMAT, 328
  RLOAD, 328
  Sn (state "n"), 328
  SXNAME, 328
  TIMESCALE, 329
  TIMESTEP, 329
digital power supplies, 322, 323, 333
digital primitives, 247
  format, 250
digital simulation
  results in files, 92
  worst-case timing, 64
digital time step, 61
digital worst-case timing, 63
  convergence hazard, 63
  cumulative ambiguity hazard, 63
digital input voltage, 63
  glitch suppression, 64
  net state conflict, 63
  persistent hazard, 64
  zero-delay-oscillation, 64
digital-to-analog
  converter, 51
  interface devices, 324
DIGMNTYMX (.OPTIONS), 61, 251
DIGMNTYSCALE (.OPTIONS), 61, 252
DIGOVRDVR (.OPTIONS), 61
DIGTYMXSCALE (.OPTIONS), 61
DINPUT device model, 51, 324
diode model, 51, 105, 108, 131
display file, xxvii
.DISTO (small-signal distortion), 102
.DISTRIBUTION, 37, 60
distribution
  user-defined, 37
  using an include file, 37
distribution name
  GAUSS, 52
  user-defined, 52
distributions
  UNIFORM tolerances name, 52
DLTCH, 270
DLYLINE, 274
doping tail, 190, 206, 348
dot command, 128, 142, 348
DOUTPUT device model, 51
drain, 71
  area, 61
  bulk, 175
  induced, 186
drive resistance, 61

E
digital simulation
  results in files, 92
  worst-case timing, 64
digital time step, 61
digital worst-case timing, 63
  convergence hazard, 63
  cumulative ambiguity hazard, 63
edge-triggered flip-flops, 51, 265
  truth tables, 268, 269
ELSE, xix, 348
emitter, 71
ENABLE, 348
.END, 39
end of circuit, 39
end subcircuit definition, 84
ENDREPEAT, 313
.ENDS, 84

environment variables
   LIBPATH, 45

equation
   changing, 339
ERRORLIMIT, 308
examples
   CONSTRAINT primitive, 309
EXPAND (.OPTIONS), 46, 59
exponential (EXP), xix
exponential temperature coefficient (TCE), 217
expressions, xx, 42
   conventions, xviii
   numeric conventions, xix
text, 88
.EXTERNAL, 40
external port specifications, 40

F
FALSE, 306
ferromagnetic, 169
Ferroxcube, 162
FET, 51
file, xxvi, 317
data, xxvi
header, 317
input, xxv
output, xxvi
stimulus, 317
transitions, 317
files
   log, xxiii, xxvi
   stimulus, 93
filter shift, 80
final time value, 90, 142
flicker noise, 126, 135, 159, 202, 214, 348
flip-flops and latches, 248, 264
   initialize, 264
   timing constraints, 264
   X-level handling, 264
flush interval, xxvi
FORMAT, 326
format array, 316
.FOUR, 41
Fourier analysis, 30, 41, 90, 348
FREQ (constraint check), 307

frequency
   expression, 136
   modulation, 143
response, AC analysis, 32
FSTIM, 86, 88, 317, 319, 348
.FUNC, 42
function definition, 42
functions
   absolute value (ABS), xix
   arc tangent (ATAN and ARCTAN), xix
   arccosine (ACOS), xix
   arctangent (ARCTAN), xix
   arsine (ASIN), xix
   ATAN2, xix
cosine (COS), xix
cosine hyperbolic, xix
differential (Ddt), xix
   exponential (EXP), xix
   hyperbolic tangent (TANH), xx
   IF, xix
   imaginary (IMG), xix
   integral (Sdt), xx
   limit (LIMIT), xix
   log base 10 (LOG10), xix
   log base E (LOG), xix
   MAX, xix
   MIN, xix
   phase (P), xix
   power (PWR), xx, xxi
   real (R), xx
   signed power (PWRS), xx
   signum (SGN), xx
   sine (SIN), xx
   square root (SQRT), xx
   step (STP), xx
   table (TABLE), xx
tangent (TAN), xx

G
GaAs MESFET, 51
GaAsFET, 105, 108, 110, 113, 343
device model, 51
illustration, 342
Level 1 parameters, 113
Level 2 parameters, 113
Level 3 parameters, 115
Level 4 parameters
parameters for all levels, 112
gate, 348
gated latch, 270
gates, 51, 71, 254
bidirectional transfer, 261
flip-flops
number in model, 258
number of inputs, 255
standard, 255
tri-state, 258
gate-source voltage, 70
Gaussian, 52
GENERAL (constraint check), 307
glitch, 348
suppression, 64
global
node names, 85
parameters, 348
ports, 348
GMIN (.OPTIONS), 61
goal functions
command line, xxvii
group delay, 73
AC analysis, 32
Gummel-Poon transistor model
quasi-saturation effect, 212

H
harmonics, 41
header
file, 317
HEX radix functions, 317
hexadecimal notation, 314
HEXFET, 203
HOLDTIME, 305
hyperbolic tangent (TANH), xx
hysteresis, 165

I
I/O model, 311
.IC, 43, 55
bias point
.IC setting, 43
IC=, 90, 128, 170
icon, xvi, 348
ideal transmission line, 224
IF, xix, 348
IGBT
extracting model parameters from data sheets, 239
imaginary part, 73
IMG(x), xix
.INC, 42, 44, 50
included file, 31, 85, 96, 348
.DISTRIBUTION command, 60
including files, 42, 44
INCR BY, 315
IND device model, 51
independent current source & stimulus, 105, 108, 142
sinusoidal waveform, 151
independent sources
AC analysis, 32
independent voltage source & stimulus, 105, 108, 142
inductor, 51, 105, 108, 170
coupling, 105, 108, 161
coupling (transformer core), 160
inductors, 216
current coefficients, 102
initial bias point condition, 43
in-line comment, 100
input file, xxv
input/output model parameters, 251, 322
AtoD, 322
DIGPOWER, 322
DRV, 322
dtoA, 322
OUT, 322
TPWRT, 322
instance name, 107, 348
instantiate, 84, 348
insulated gate bipolar transistor (IGBT), 105, 108, 237
integral (Sdt), xx
Intel hex format, 279, 283
interface, 246
internal time step, 91
INTERNAL_NODE, 344
INV, 256

358
INV3, 259
invoke, 349
IO_LEVEL, 251, 311
IO_STM, 311, 313, 320
ionization knee, 157, 349
IS =, 326
IS temperature, 112, 133, 155, 208, 349
ISAS, 348
ISWITCH device model, 51
iteration, 149, 349
ITL3, 102
ITLn (.OPTIONS), 61

J
JEDEC file, 86, 88, 275, 277, 278, 334
JFET, 105, 108, 153, 343
Jiles-Atherton model, 163, 165, 349
JKFF, 265
job statistics (ACCT), 59
junction
definition, 175, 178, 349

K
KBREAK (inductor coupling), 163
KEYWORD, 136, 148, 349
DEG, 137
MAG, 137
PARAMS, 236
R_I, 137
RAD, 137
Knuth, Donald, 48

L
labels, 349
Laplace variable, 42, 136
latch, 264
gated, 270
lateral PNP, 51
.LIB, 45
LIBPATH, 45, 349
LIBRARY (.OPTIONS), 59
library file, 45
limit (LIMIT), xix
LIMPTS (.OPTIONS), 62
LIMTIM, 102
linear
temperature coefficient (TC1), 217
link, 345, 349
LIST (.OPTIONS), 59
.LOADBIAS, 76
load bias point file, 46
log (logarithmic)
base 10 (LOG10), xix
base E (LOG), xix
log files, xxiii
PSpice, xxvi
log Probe commands, xxiii
logic expression, 292
LOGICEXP, 292, 293, 295
lossy transmission line, 51, 225
lot tolerance, 52, 96, 98, 349
LPNP device model, 51
lsb, 93, 349
LVLCOD, 102
LVLTIM, 102

M
M.H, 337
macro file, xxvii
MAG, 137
magnetic core, 161, 163
magnitude, 62, 73
M(x), functions, xix
master library file, 45
MAXFREQ, 307
maximum
(MAX), xix
MAXORD, 102
.MC, 47
memory primitives, 249
RAM, 283
ROM, 279
messages, 63, 308
DIGITAL INPUT VOLTAGE, 63
FREQUENCY, 63
GENERAL, 63
hazard and timing violation, 63
HOLD, 63
NET-STATE CONFLICT, 63
PERSISTENT HAZARD, 64
persistent hazards, 40
RELEASE, 63
SETUP, 63
Timing Violations, 63
WIDTH, 63
ZERO-DELAY-OSCILLATION, 64

metafile, 349
METHOD, 102
Microsoft compiler for the device equations option, 345
MIN_LO, 306
MINFREQ, 307
minimum (MIN), xix
MNTYMXDLY, 251
mobility, 181, 185, 186, 191, 240, 241, 349
.MODEL, 50
model
library, 45, 349
temperature customization, 53
model breakout parts
resistors, 216
D (diode), 132
GASFET (GaAsFET), 111
IGBT, 238
ISWITCH (current-controlled switch), 233
LPNP (bipolar transistor), 205
NJF (JFET), 154
NMOS (MOSFET), 177
NPN (bipolar transistor), 205
PJF (JFET), 154
PMOS (MOSFET), 177
PNP (bipolar transistor), 205
VSWITCH (voltage-controlled switch), 220
X (diode), 132
models, using in circuit designs, 106
Monte Carlo analysis, 37, 47, 76
default distribution values, 60
read-in error, 80
MOS.C and device model changes, 337
MOS.C and state vector information, 341
MOSFET, 105, 174
BSIM3 version 3.1 model description, 181
BSIM3 version 3.1 model parameters, 192
device declaration, 108
EKV version 2.6 model description, 179
EKV version 2.6 model parameters, 187
Level 4 description, 178
Levels 1, 2, and 3 descriptions, 178
model declaration, 51
model parameters, 178
mouse, 349
msb, 93, 349
msim.ini, 45, 349
multi-bit A/D converter, 249, 286
timing model, 287
multi-bit D/A converter, 249, 286, 289
timing model, 289
MXPR macro, 338

N
N device, 324
Nagel, Lawrence, 339
NAND, 256
NAND3, 259
NBTG, 261
N-channel, 51
GaAsMESFET, 51
IGBT, 51
JFET, 51
NMOS, 51
nested subcircuits, 102
nesting, 236, 349
netlist
definition, 349
device declarations, 107
intrinsic device types, 107
proper syntax, 106
NIGBT device model, 51
NJF device model, 51
NMOS device model, 51
NOBIAS (.OPTIONS), 59
NODE (.OPTIONS), 59, 306
.NODESET, 46, 55, 76
nodeset, 30, 46, 55, 77, 349
NOECHO (.OPTIONS), 59
.NOISE, 56
noise
analysis, 56
bipolar transistor, 214
diode, 135
flicker, 126, 135, 159, 202, 214
JFET, 159
MOSFET, 202
shot, 126, 159, 202, 214
thermal, 126, 135, 159, 202, 214, 235
NOM.LIB, 45
nominal temperature, 87, 117, 130, 134, 156, 173, 198, 209, 218
NOMOD (.OPTIONS), 60
nonlinear controlled sources, 58
NOOUTMSG (.OPTIONS), 60
NOPAGE (.OPTIONS), 60
NOR, 256
NOR3, 259
NOREUSE (.OPTIONS), 60
NOREUSE flag, 60, 77, 349
NOSUBCKT, 75, 350
NPN bipolar transistor, 51
number of times to repeat (n), 312
NUMDGT (.OPTIONS), 62, 68, 350
numeric expression convention, xix
NXOR, 256
NXOR3, 259

O
O device, 328
OA, 256
object, 345, 350
OCT radix functions, 317
OMITTED, 339
.OP, 58
operator, 88, 350
.OPTIONS, 59, 61
  ABSTOL, 61
  CHGTOL, 61
  CPTIME, 61
  DEFA, 61
  DEFA, 61
  DEFL, 61
  DEFW, 61
  DIGDRVF, 61
  DIGDRVZ, 61
  DIGERRDEFAULT, 308
  DIGERRLIMIT, 308
  DIGFREQ, 61
  DIGINITSTATE, 61
  DIGOLVL, 61
  DIGMNTYM, 61
  DIGMNTYS, 252
  DIGOV, 61
  DIGTVM, 61
  DIGTVMS, 61
  DISTRIBUTION, 60
  EXPAND, 46
  GMIN, 61
  ITL1, 61
  ITL2, 61
  ITL4, 61
  ITL5, 61
  LIBRARY, 59
  LIMPTS, 62
  LIST, 59
  NOBIAS, 59
  NODE, 59
  NOECHO, 59
  NOMOD, 60
  NOOUTMSG, 60
  NOPAGE, 60
  NOPRBMSG, 60
  NOREUSE, 60
  NUMDGT, 68
  OPTS, 60
  PIVREL, 62
  PIVTOL, 62
  RELTOL, 62
  STEPGMIN, 60
  TNOM, 62
  VNOM, 62
  WIDTH, 68
  ACCT, 59
options, xxv, 59
options not available in PSpice, 102
MAXORD, 102
.OPTS (.OPTIONS), 60
OR, 256
OR3, 259
OUTPUT ALL, 47, 95, 96, 350
output file, xxvi
  and flag options, 59
  resistances, 89
output files
  digital simulation results, 92
output variables
  DC sweep, 70

361
transient analysis, 70

P
package, 350
page options for simulation output, 60
page, definition, 350
.PARAM, 65
parameter, 350
  adding with the device equations option, 338
definition, 65
global, 112, 130, 133, 348
  name change using the device equations option, 338
value calculation using the device equations options, 339
parametric analysis, 79
PARAMS, 84
  subcircuits, 236
part, 350
definition, 350
instance, 350
path definition, 296
PBTG, 261
P-channel, 51
  JFET, 51
  MOSFET, 51
peak amplitude, 150, 151
phase, 73
phase (P), xix
piecewise linear, 37
pin, 215, 291, 333, 350
pin current, 350
pin names equivalent to node names, 33
pins
  and capacitor node polarity, 128
  and inductor node polarity, 170
  as external interface points of networks, 40
  subcircuit nodes, 84
  voltage out of range, 63
pin-to-pin delay (PINDLY), 294, 296, 298, 303
PIVREL (.OPTIONS), 62
PIVTOL (.OPTIONS), 62
PJF device model, 51
PLAND, 276
PLD, 86, 88, 275
  overview, 275
PLNAND, 276
PLNOR, 276
PLNXOR, 276
PLOR, 276
.PLOT, 66
plot, 66
PLXOR, 276
PMOS device model, 51
PNP device model, 51
POLY, 350
  polynomial transfer function (for ABM controlled sources), 138
ports, 350
global, 348
power (PWR), xx, xxi
.prb file, xxvii
primitives, 246, 247, 295
.PPRINT, 68
print, 68
  step value, 90, 142
tables, 68
Probe, 69
data files, 58
output files, 39
.PROBE, 69
Probe command line options
  -c, xxvi
  -p, xxvii
programmable logic array PLD, 334
  data values, 277
  overview, 275
PLAND, 276
PLANDC, 276
PLNAND, 276
PLNANDC, 276
PLNOR, 276
PLNORC, 276
PLNXOR, 276
PLNXORC, 276
PLOR, 276
PLORC, 276
PLXOR, 276
PLXORC, 276
syntax, 276
timing model, 278
types, 248, 276
propagation delay, 252
PSpice log file, xxvi
PSpice command line options
-d, xxvi
-l, xxvi
-o, xxvi
PSpice messages
hazard and timing violations, 63
Persistent Hazards, 40
PSPICE.MAK, 345
PULLDN, 273
PULLUP, 273
pullup and pulldown resistors, 248, 273

Q
QBAR, 69
quadratic temperature coefficient (TC2), 217
quasi-saturation effect (BJT), 212

R
R device, 215
R_I, 137
RAD, 137
RAM, 283
random access memory
timing model, 284
read only memory, 279
timing model, 282
real function (R), xx
real part, 73
recompiling and linking, 345
reference functions, 298
CHANGED, 298
CHANGED_HI, 298
CHANGED_LH, 298
relative accuracy, 62
RELEASETIME, 305
RELTOL (.OPTIONS), 62
REPEAT, 149, 313
ENDREPEAT, 149
FOREVER, 149
RES device model, 51
resistance multiplier, 217
resistor, 51, 105, 108, 215
model definition, 215
pullup and pulldown, 273
RMS, 56
roll-off current, 206
ROM, 279
run, 137, 219, 350
Probes commands, xxiii

S
S device, 219
save
.bias point to file, 75
.SAVEBIAS, 46, 75
scale factor, 61
SCBE, 181, 190, 350
schematic, 350
Schmitt trigger, 322
Sdt(x) integral function, xx
semiconductor models, 103
semiconductor parts, 132, 177
.SENS, 78
sensitivity
analysis, 78
worst-case analysis, 95
setpoint, 350
setting initial conditions, 43
SETUP_HOLD
constraint check, 304
SETUPTIME, 305
SGN(X) signum function, xx
shot noise, 126, 159, 202, 214
shunt conductance, 65
SIGNAME, 320, 327
signed power (PWRS), xx
SIMLIBPATH, 351
simulation, 137, 225, 351
lossy transmission lines, 225
transmission line, 223
simulation command line options
-bf, xxvi
-bn, xxvi
-bs, xxvi
-e, xxvi
-i, xxvi
-wDAT, xxvii
-wNO_NOTIFY, xxvii
KCOUPLEn (transmission line coupling matrix), 227
L (inductor), 171
LBREAK (inductor), 172
MBREAKx (MOSFET), 177
QBREAKx (bipolar transistor), 205
R (resistor), 216
R_VAR (resistor), 216
RBREAK (resistor), 216
SBREAK (voltage-controlled switch), 220
T (transmission line), 226
TLOSSY (transmission line), 226
TnCOUPLED (transmission line), 227
TnCOUPLEDX (transmission line), 227
WBREAK (current-controlled switch), 233
XFRM_LINEAR (transformer), 163, 171
XFRM_NONLINEAR (nonlinear transformer), 163
ZBREAKN (IGBT), 238

syntax, xvii, 52, 83, 351

T
device, 223
T_ABS (.MODEL), 53
T_MEASURED (.MODEL), 53
T_REL_GLOBAL (.MODEL), 53
T_REL_LOCAL (.MODEL), 53
table (TABLE), xx
tangent (TAN), xx
tangent hyperbolic (TANH), xx
tC, 217
  1 (linear temperature coefficient), 217
  2 (quadratic temperature coefficient), 217
  E (exponential temperature coefficient), 217
.TEMP, 87
temperature, 65, 87
customization, 53
temperature customizing, 53
.TEXTURE, 87
.text expressions, 88
text parameter definition, 88
.TEXTEXTINT, 88, 351
.TF, 89
thermal noise, 126, 135, 159, 202, 214, 235
tick number, 326, 351
TIMESCALE, 317, 351
TIMESTEP, 312, 314, 326, 330
timing constraint, 253
timing hazards
  convergence, 63
  cumulative ambiguity, 63
timing model, 251
delay line, 274
gated latch, 270
general discussion, 252
multi-bit A/D converter, 287
multi-bit D/A converter, 289
programmable logic array, 278
random access memory, 284
read only memory, 282
TNOM (.OPTIONS), 62
tolerances, 52, 98
DEV, 52
distribution name, 52
GAUSS, 52
LOT, 52
specification, 52
UNIFORM, 52
user-defined, 52
TOM model, 110, 351
topology, 77, 342
.TRAN, 90
transient function, 89
transformer, 51
transient analysis, 41, 90
  final time value
  internal time step, 91
  no-print value, 90
  print step value, 90
SKIPBP, 90
step ceiling value, 90
transient bias point, 43
transistor, bipolar, 204
transition functions, 299
TRN_$H, 299
TRN_$L, 299
TRN_HS, 299
TRN_HL, 299
TRN_HZ, 299
TRN_L$, 299
traces

TRN_LH, 299
TRN_LZ, 299
TRN_ZS, 299
TRN_ZH, 299
TRN_ZL, 299

transitions

file, 317

transmission line

ideal line, 224
illustration, 224, 225
lossy line, 225

transmission line coupling, 105, 109, 160, 167, 223

transmission line device, 223

transmission lines

coupled, 228

ideal, 226
lossy, 226
simulating, 228

TRISTATE, 296, 301

tri-state gates, 51, 248, 258, 259

AND3, 259
AND3A, 259
BUF3, 259
BUF3A, 259
INV3, 259
INV3A, 259
NAND3, 259
NAND3A, 259
NOR3, 259
NOR3A, 259
NXOR3, 259
NXOR3A, 259
OR3, 259
OR3A, 259
XOR3, 259
XOR3A, 259

TRN device model, 51

TSTEP, 142

TSTOP, 142

typographical conventions, xvi

U

U.C. Berkeley, 337
address, 203, 339
UADC device model, 51, 287
UBTG, 261

UDAC device model, 51, 289
UDLY device model, 51, 274
UEFF device model, 51, 265
UGATE, 293
UGATE device model, 51, 255, 292
UGFF device model, 51, 270
UIO device model, 51, 322
UPLD, 275
URAM, 283
UROM, 279
user-defined distribution, 37

UTGATE device model, 51, 258

V

VARY BOTH, 96, 351

VARY DEV, 95, 351

VARY LOT, 96, 351

.VECTOR, 92

VIEWsim A/D, 327

VNTOL (.OPTIONS), 62

Voltage Source

PWL Parameters, 148
voltage-controlled

current source, 105, 107, 136
switch, 51, 105, 109, 219

voltage source, 105, 107, 136

VSWITCH device model, 51

VTO temperature, 351

W

W device, 232

.WATCH, 94

watch analysis results, 94

.WCASE, 95

WHEN, 308

.WIDTH, 102

IN= option, 102

WIDTH (.OPTIONS), 62, 68, 99

WIDTH (constraint check), 306

wildcard characters, xxv

window menu, 351

Windows, xvi, 63

worst-case analysis, 37, 76, 95
X
X devices, 84
XOR, 256
XOR3, 259

Z
zero impedance voltage source, DAC, 289